

Marcus Grum

Construction of a Concept of Neuronal Modeling

MOREMEDIA



Springer Gabler

Gabler Theses

In der Schriftenreihe „Gabler Theses“ erscheinen ausgewählte, englischsprachige Doktorarbeiten, die an renommierten Hochschulen in Deutschland, Österreich und der Schweiz entstanden sind. Die Arbeiten behandeln aktuelle Themen der Wirtschaftswissenschaften und vermitteln innovative Beiträge für Wissenschaft und Praxis. Informationen zum Einreichungsvorgang und eine Übersicht unserer Publikationsangebote finden Sie hier.

Weitere Bände in der Reihe <https://link.springer.com/bookseries/16768>

Marcus Grum

Construction of a Concept of Neuronal Modeling

Marcus Grum
Berlin, Germany

ISSN 2731-3220

Gabler Theses

ISBN 978-3-658-35998-0

<https://doi.org/10.1007/978-3-658-35999-7>

ISSN 2731-3239 (electronic)

ISBN 978-3-658-35999-7 (eBook)

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Fachmedien Wiesbaden GmbH, part of Springer Nature 2022

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Responsible Editor: Marija Kojic

This Springer Gabler imprint is published by the registered company Springer Fachmedien Wiesbaden GmbH part of Springer Nature.

The registered company address is: Abraham-Lincoln-Str. 46, 65189 Wiesbaden, Germany

*“What is stronger than the human brain
which shatters over and over and still
learns.”*

in the style of Rupi Kaur

Acknowledgements

Following the moral thinking example of Hare, I offer this research to the public now rather than later (Hare, 1981). This is not because I think it needs no improvement but because of a sense of relevance. It is based on a feeling that if these ideas are understood, researchers and practitioners might progress in addressing practical issues and the major issues of modern society.

I have had to leave many questions unresolved, such as the identification of a satisfactory, complete set of tools for neuronal process modeling. Since I recognize the fact that I cannot realistically hope to write the last word on a scientific discourse and the most I can do is progress scientific discussions, I hope that I have enabled some things to be made clearer.

Too many people have helped me to be mentioned here. Just naming them would fill most of the page. But I would like to thank some who have been kind enough to look at the transcripts, offer comments and willing discuss questions. This includes Prof. Dr.-Ing. Norbert Gronau in particular, and also Prof. Dr. Tobias Scheffer, who were patient supervisors, and Dr. Edzard Weber, a conscientious mentor. They might not agree with what I have done in response, but I owe a lot to them. I am grateful to have obtained the freedom to work on this topic.

Besides those I have already mentioned, I must include Dr. Wendelin Böhmer, Dr. Michael Scholz, Prof. Dr. Klaus Obermayer, Prof. Dr. Oliver Brock and Prof. Dr. Attahiru S. Alfa, who have accompanied my first steps in research; Prof. Dr. Moreen Heine and Prof. Dr.-Ing. Ingo Timm because of their valuable feedback on my work; Prof. Dr.-Ing. Milos Kristic, Prof. Dr.-Ing. Sebastian Reich, Prof. Dr.-Ing. Christian Hammer and Prof. Dr. Andreas Schwill have trustfully guided the last steps of this dissertation; as well as Prof. Dr. Sepp Hochreiter and Prof.

Dr. Jürgen Schmidhuber for inspiring conversations and contributions throughout the years.

The highly inspiring environment created at the University of Potsdam and the Department of Business Informatics, especially Processes and Systems, as well as its research environment, have proven to be fruitful and appropriate. I owe thanks to my colleagues for their stimulating talks throughout my studies. Beside the ones I already have mentioned, I must include Dr. Gergana Vladova, Dr. André Ullrich, Dr. Sander Lass, Dr. Christof Thim, Dr. Eldar Sultanow, Hanna Theuer, David Kotarski and, of course, Benedict Bender. I owe more thanks to the director of my department, Prof. Dr.-Ing. Norbert Gronau, for constant support, unremitting kindnesses and providing the necessary materials, such as AR techniques, work machines and conference resources.

Above all, I would like to express my gratitude to my wife not only for her patience while challenging my perspectives and explanations and checking my text but rather for keeping me sane during the writing process. Particularly her patience while I worked during the evenings, weekends and holidays has prepared the path for writing this research.

What convinces me more than anything of the importance and correctness of the account of neuronal process modeling presented is the fact that this offers more explanation than any other method I have seen, and this will practically help to clarify contemporary issues. Since my aim in pursuing neuronal process modeling, which includes simulation and optimization, has always been to discover the world's inner relations and what holds the world together, I dedicate this research to all who feel the need for such clarification and those who responsibly and morally deal with it.

Public CoNM repository: <https://github.com/MarcusGrum/CoNM>

Berlin

2020

Marcus Grum

Abstract

English: The business problem of having inefficient processes, imprecise process analyses and simulations as well as non-transparent artificial neuronal network models can be overcome by an easy to use modelling concept. With the aim of developing a flexible and efficient approach to modeling, simulating and optimizing processes, this paper proposes a flexible Concept of Neuronal Modeling (CoNM). The modelling concept, which is described by the modeling language designed and its mathematical formulation and is connected to a technical substantiation, is based on a collection of novel sub-artifacts. As these have been implemented as a computational model, the set of CoNM tools carries out novel kinds of Neuronal Process Modeling (NPM), Neuronal Process Simulations (NPS) and Neuronal Process Optimizations (NPO). The efficacy of the designed artifacts was demonstrated rigorously by means of six experiments and a simulator of real industrial production processes.

German: Die vorliegende Arbeit adressiert das Geschäftsproblem von ineffizienten Prozessen, unpräzisen Prozessanalysen und -simulationen sowie untransparenten künstlichen neuronalen Netzwerken, indem ein Modellierungskonzept zum Neuronalen Modellieren konstruiert wird. Dieses neuartige Konzept des Neuronalen Modellierens (CoNM) fungiert als flexibler und effizienter Ansatz zum Modellieren, Simulieren und Optimieren von Prozessen mit Hilfe von neuronalen Netzwerken und wird mittels einer Modellierungssprache, dessen mathematischen Formalisierung und technischen Substanzierung sowie einer Sammlung von neuartigen Subartefakten beschrieben. In der Verwendung

derer Implementierung als CoNM-Werkzeuge können somit neue Arten einer Neuronalen-Prozess-Modellierung (NPM), Neuronalen-Prozess-Simulation (NPS) sowie Neuronalen-Prozess-Optimierung (NPO) realisiert werden. Die Wirksamkeit der erstellten Artefakte wurde anhand von sechs Experimenten demonstriert sowie in einem Simulator in realen Produktionsprozessen gezeigt.

Contents

1	Introduction	1
1.1	Initial Situation	1
1.2	Research Aims and Objectives	6
1.3	Proposed Research Design	8
1.4	Structure	10
2	Problem Analysis and Problem Statement	13
2.1	Process Modeling	14
2.1.1	Basic Concepts and Definitions	14
2.1.1.1	Models	15
2.1.1.2	Meta-Models	25
2.1.1.3	Processes and Business Processes	27
2.1.1.4	Modeling	38
2.1.2	Process Taxonomies	42
2.1.2.1	Perspective-Oriented Taxonomies	42
2.1.2.2	Approach-Oriented Taxonomies	46
2.1.2.3	Architecture-Based Taxonomies	47
2.1.2.4	Critical Appraisal of Taxonomies	53
2.1.3	Process Modeling Languages	55
2.1.3.1	Petri Nets	57
2.1.3.2	(e)EPC	64
2.1.3.3	UML 2's Activity Diagram	68
2.1.3.4	BPMN	75
2.1.3.5	Critical Appraisal of PMLs	81
2.1.4	Conclusion About Taxonomies and PMLs	83

2.2	Knowledge Modeling	85
2.2.1	Basic Concepts and Definitions	86
2.2.1.1	Learning and Knowledge	86
2.2.1.2	Knowledge Representation Formalisms	103
2.2.1.3	Knowledge Transfers	107
2.2.2	Knowledge Management Concepts	112
2.2.2.1	Promote	115
2.2.2.2	GPO-WM	120
2.2.2.3	Potsdam KM Model	124
2.2.2.4	Critical Appraisal of KMCs	129
2.2.3	Knowledge Modeling Languages	131
2.2.3.1	Promote	134
2.2.3.2	GPO-WM	138
2.2.3.3	KMDL	143
2.2.3.4	Critical Appraisal of KMLs	150
2.2.4	Conclusion About KMCs and KMLs	151
2.3	Process Simulation	154
2.3.1	Basic Concepts and Definitions	154
2.3.1.1	Methods of Simulations	155
2.3.1.2	Simulation Systems	159
2.3.1.3	Simulation and Process Simulations	160
2.3.1.4	Simulating	162
2.3.2	Simulation Taxonomies	170
2.3.2.1	Behavior-oriented Taxonomies	172
2.3.2.2	Presence of Time-oriented Taxonomies	173
2.3.2.3	Basis of Value-oriented Taxonomies	175
2.3.2.4	Feedbacking-oriented Taxonomies	176
2.3.2.5	Termination-oriented Taxonomies	176
2.3.2.6	Stationarity-oriented Taxonomies	177
2.3.2.7	Real-Time-oriented Taxonomies	177
2.3.2.8	Symbiosis-oriented Taxonomies	178
2.3.2.9	Digitalization-oriented Taxonomies	179
2.3.2.10	Time Advancement-oriented Taxonomies	179
2.3.2.11	Specialization-oriented Taxonomies	180
2.3.2.12	Critical Appraisal of Taxonomies	182
2.3.3	Simulation Modeling Languages	184
2.3.3.1	Space Markup Language	186
2.3.3.2	Flowcharts	190

2.3.3.3	System Dynamics	195
2.3.3.4	Critical Appraisal of SMLs	201
2.3.4	Conclusion About Taxonomies and SMLs	203
2.4	Process Optimization	205
2.4.1	Basic Concepts and Definitions	206
2.4.1.1	Optimizing Models	206
2.4.1.2	Learning Organizations	211
2.4.2	Optimization Philosophies	218
2.4.2.1	Business Process Reengineering	220
2.4.2.2	Continuous Improvement Processes	224
2.4.2.3	Reference Modeling	226
2.4.2.4	Critical Appraisal of Philosophies	230
2.4.3	Organizational Learning	231
2.4.3.1	Single-Loop Learning	233
2.4.3.2	Double-Loop Learning	235
2.4.3.3	Deutero-Learning	236
2.4.3.4	Critical Appraisal of Organizational Learning	238
2.4.4	Conclusion About POPs and Organizational Learning	240
2.5	Deep Learning with Neuronal Networks	242
2.5.1	Biological Motivation	243
2.5.1.1	The Nervous System	243
2.5.1.2	The Brain	244
2.5.1.3	The Neuron	245
2.5.1.4	The Signal Transmission	246
2.5.1.5	Further Principles	247
2.5.2	Learning Tasks	249
2.5.2.1	Fixed Learning Task	250
2.5.2.2	Free Learning Task	251
2.5.2.3	Reinforced Learning Tasks	251
2.5.3	Knowledge Codification and Encoding	252
2.5.4	Learning Problems	253
2.5.4.1	Cluster Problems	254
2.5.4.2	Classification Problems	256
2.5.4.3	Prediction Problems	257
2.5.5	Neuronal Networks	259
2.5.5.1	Single Neurons	260
2.5.5.2	Multilayer Perceptrons	263

2.5.5.3	Recurrent Neuronal Networks	265
2.5.6	Training of Neuronal Networks	267
2.5.6.1	Error-based Learning Algorithms	267
2.5.6.2	Competitive Learning Algorithms	271
2.5.6.3	Hebbian Learning Algorithms	272
2.5.6.4	Target Functions	273
2.5.6.5	Weight Initialization	276
2.5.6.6	Input Representation	277
2.5.6.7	Generalization	278
2.5.7	Network Interpretability and ANN Explainability	279
2.5.7.1	Sensitivity Analyses	279
2.5.7.2	Feature Visualization	280
2.5.7.3	Saliency	283
2.5.7.4	Neuronal Fuzzy Systems	286
2.5.8	Network Architectures	294
2.5.8.1	Radial Basis Networks	294
2.5.8.2	Partially Recurrent Networks	298
2.5.8.3	Long-Short-Term-Memory Blocks	300
2.5.8.4	Convolutional Networks	310
2.5.8.5	Topological Networks	313
2.5.8.6	Auto-associative Networks	315
2.5.9	ANN Modeling	320
2.5.9.1	TensorFlow Graphs	321
2.5.9.2	NeuroML	326
2.5.9.3	LEMS	330
2.5.9.4	Critical Appraisal of ANN MLs	334
2.5.10	Conclusion About Deep Learning	336
2.6	Software Tooling	339
2.6.1	Modeling and Simulation Tools	340
2.6.1.1	Selection of Adequate Tools	341
2.6.1.2	ARIS Platform	352
2.6.1.3	ADONIS	355
2.6.1.4	Modelangelo	358
2.6.1.5	Signavio Process Manager	361
2.6.1.6	AnyLogic	364
2.6.1.7	SimPy	367
2.6.1.8	MATLAB	368
2.6.1.9	Critical Appraisal	374

2.6.2	ANN Tools	377
2.6.2.1	Selection of Adequate Tools	378
2.6.2.2	PyBrain	385
2.6.2.3	Theano	387
2.6.2.4	Caffe(2)	390
2.6.2.5	TensorFlow	393
2.6.2.6	NeuroConstruct	395
2.6.2.7	NeuroLOTs and NeuroTessMesh	397
2.6.2.8	NEURON	399
2.6.2.9	GENESIS	403
2.6.2.10	Critical Appraisal	408
2.6.3	Conclusion About Tools and Libraries	411
2.7	Problem Analysis and Research Gap	412
3	Objectives and Methodology	429
3.1	Analysis of Objectives	430
3.1.1	CoNM Terminology	430
3.1.2	Classification and Limitation of the Object of Reflection	434
3.1.3	Requirements	436
3.2	Methodological Additives	442
3.2.1	Method for the Systematic Literature Review	443
3.2.2	Design Science Research Methodology	449
3.2.3	Method for the Development of Neuronal Networks ...	452
3.2.4	Empirical Evaluation Method	454
3.2.5	Morphological Box	456
3.2.6	Mixed Methods Design	456
3.3	Methodical Approach	456
4	Design	461
4.1	Taxonomy and Meta-Model Creation	463
4.1.1	ANN Process Taxonomy	464
4.1.1.1	Business Process Context Perspective	467
4.1.1.2	Behavioral Perspective	469
4.1.1.3	Functional Perspective	470
4.1.1.4	Informational Perspective	472
4.1.1.5	Knowledge Perspective	473
4.1.1.6	Organizational Perspective	474
4.1.1.7	Neuronal Perspective	475
4.1.1.8	Process ANN Perspective	477
4.1.1.9	Simulation Perspective	478

4.1.2	Specification of Object Models	479
4.1.2.1	Object Conceptualization	481
4.1.2.2	Relationship Modeling	481
4.1.2.3	Integration of Objects and Relationships	483
4.1.3	Meta-Model	484
4.1.4	Interim Conclusion	486
4.2	Generic Evaluation and Selection System	486
4.2.1	Subset Definition	488
4.2.2	Evaluation Task Definition	490
4.2.3	Visualization Design	492
4.2.4	Evaluation Aspects Concretion	493
4.2.5	Evaluation Aspect Operationalization	493
4.2.6	Object of Investigation Concretion	494
4.2.7	Interim Conclusion	495
4.3	Neuronal Enterprise Architecture	495
4.3.1	NEA Structure	496
4.3.2	NEA Associations	499
4.3.3	NEA Phases and Roles	500
4.3.4	Interim Conclusion	500
4.4	CoNM Proceeding	501
4.4.1	CoNM Proceeding Phases	502
4.4.2	CoNM Proceeding Associations	505
4.4.3	Interim Conclusion	505
4.5	Neuronal Knowledge Management Model	506
4.5.1	Activity Collection	508
4.5.2	Educative Range Specification	513
4.5.3	Organizational Range Specification	514
4.5.4	Procedural Range Specification	516
4.5.5	Personnel Range Specification	517
4.5.6	Framework Conditions	518
4.5.7	NKMM Ordering System	520
4.5.8	Interim Conclusion	521
4.6	Concept of Neuronal Process Modeling	522
4.6.1	System Specification	525
4.6.1.1	Network Systems	526
4.6.1.2	Process Systems	528
4.6.1.3	Activity Systems	530
4.6.1.4	Activation Systems	532
4.6.2	Generic View Specification	535

4.6.3	Syntax and Convention Specification	539
4.6.3.1	Simulation Views	541
4.6.3.2	Business Process Context Views	546
4.6.3.3	Behavioral Views	552
4.6.3.4	Functional Views	558
4.6.3.5	Process ANN Views	563
4.6.3.6	Knowledge Views	572
4.6.3.7	Organizational Views	578
4.6.3.8	Informational Views	585
4.6.3.9	Neuronal Views	593
4.6.4	Modeling Shape Specification	599
4.6.5	Interim Conclusion	606
4.7	Systematic Exhaustion of Knowledge Objects	607
4.7.1	Static Knowledge Identification	608
4.7.2	Dynamic Knowledge Identification	609
4.7.3	Interim Conclusion	610
4.8	Learning Principles in CoNM	611
4.8.1	Principle of Neuronal Recursion	612
4.8.2	ANN-based Simulation Systems	612
4.8.3	NPO Backpropagation	613
4.8.4	Interim Conclusion	614
5	Implementation	615
5.1	Usability Concept Realization	615
5.1.1	Bidirectional Modeling in XR	616
5.1.2	XR Modeling Use Cases	618
5.1.3	Interim Conclusion	621
5.2	Architectural Set Up	621
5.2.1	Devices	623
5.2.2	Application Servers	623
5.2.3	Database Servers	623
5.2.4	File Servers	624
5.2.5	Third Party Systems	624
5.2.6	Interim Conclusion	624
5.3	Software Prototypes	625
5.3.1	Augmentor Development	625
5.3.2	Modelangelo Extension	629
5.3.3	Transformation Tool Development	636
5.3.4	PyBrain Extension	636
5.3.5	Interim Conclusion	637

6 Demonstration	639
6.1 Identification of an Appropriate Technical Foundation	639
6.1.1 Problem Specification	640
6.1.2 Survey Planning and Preparation	640
6.1.3 Data Collection	643
6.1.4 Data Analysis	643
6.1.4.1 Personal Expertise	644
6.1.4.2 Expert View	646
6.1.4.3 Popularity and Attractiveness	650
6.1.5 Reporting	655
6.2 Neuronal Knowledge Modeling Language	656
6.2.1 NMDL Modeling Neuronal Conversions	657
6.2.2 NMDL Modeling ANN Architectures	660
6.2.2.1 Case of Single Neurons	661
6.2.2.2 Case of Multilayer Perceptrons	661
6.2.2.3 Case of Recurrent Neuronal Networks	662
6.2.2.4 Case of Long-Short-Term-Memory Blocks	663
6.3 Rise of Transparency	665
6.3.1 Experiment 1—Live Learning	665
6.4 Tacit Knowledge Identification	673
6.4.1 Experiment 2—Static Knowledge Identification	674
6.4.2 Experiment 3—Dynamic Knowledge Identification	686
6.5 Neuronal Process Simulation	693
6.5.1 Experiment 4—Time-Oriented Process Simulation	694
6.6 Neuronal Process Optimization	703
6.6.1 Experiment 5—NPO Backpropagation Procedures	703
6.6.2 Experiment 6—Standard Backpropagation Procedures	711
6.7 Conclusion about Demonstrations	715
7 Evaluation	719
7.1 Findings	721
7.1.1 Selections on Behalf the CoNM-SESS	721
7.1.2 Sub-Artifact-Specific Evaluation	726
7.1.3 Evaluations on Behalf the CoNM-SESS	731
7.2 Evaluation of the Practical Applicability	739
7.3 Attempts For Coheratism & CoNM-Contributions	762

8 Concluding Remark	767
8.1 Summary	767
8.2 Critical Appraisal	770
8.2.1 Research Questions	771
8.2.2 Methodology	774
8.2.3 Overall Result	775
8.2.4 Single Results	779
8.3 Outlook	780
Bibliography	783

Abbreviations

ADM	Architecture Development Method (method of TOGAF)
AI	Artificial Intelligence
ANN	Artificial Neural Network
ANN ML	Artificial Neuronal Network Modeling Language
AR	Augmented Reality
ARIS	Architecture of Integrated Systems
BP	Business Process
BPD	Business Process Diagram
BPMI	Business Process Management Initiative
BPMN	Business Process Modeling Notation
BPR	Business Process Reengineering
BSP	Business System Planning (1 st historic category of EA)
C/E net	Condition/Event net (a Petri net variant)
CAM	Class Activation Map (an attribution-oriented visualization method)
CASE	Computer-Aided Software Engineering
CEE	Cross Entropy Error E_{CE}
CF	Certainty Factor
CIM	Computer Integrated Manufacturing Systems
CIMOSA	Computer Integrated Manufacturing Open System Architecture
CIP	Continuous Improvement Process
CLD	Causal Loop Diagram (as part of System Dynamics)
CNPM	Concept of Neuronal Process Modeling
CNN	Convolutional Neuronal Network
CoNM	Concept of Neuronal Modeling
CPN	Colored Petri Net (a Petri net variant)

CPPS	Cyber-Physical Production System
CPS	Cyber-Physical System
CRD	Center for Reviews and Dissemination
CS	Computer Science
DARE	Database of Abstract of Reviews of Effects
DBM	Deep Boltzmann Machine
DDI	Denotation, Demonstration and Interpretation Method
DKI	Dynamic Knowledge Identification (as part of SEKO)
DL	Deep Learning, in regard with artificial neuronal networks
DLL	Double-Loop Learning
DTD	Deep Tailor Decomposition
EA	Enterprise Architecture
EAM	Enterprise Architecture Management
ECM	Enterprise Content-Management
EEA	Early Enterprise Architectures (2 nd historic category of EA)
(e)EPC	(extended) Event-driven Process Chain
ERM	Entity-Relationship-Model
ERP	Enterprise Resource Planning
Exp.	Experiment
FMEA	Failure Mode and Effects Analysis
FMECA	Failure Mode and Effects and Criticality Analysis
GAP	Global Average Pooling (method used in CNN)
GESS	Generic Evaluation and Selection System
GPO-WM	Business Process-oriented Knowledge Management (German: Geschäfts prozess orientiertes Wissens management)
GPU	Graphical Processing Unit
Grad-CAM	Gradient-weighted Class Activation Map (an attribution-oriented visualization method)
IBM	International Business Machines Corporation
i.i.d.	independent and identically distributed
IPMA	International Project Management Association
IS	Information System
ISA	Information System Architecture
IT	Information Technology
ITIL	Information Technology Infrastructure Library
IUM	German method of integrated enterprise modeling
KL	Knowledge Logistics
KM	Knowledge Management
KMC	Knowledge Management Concept

KMDL	Knowledge Modeling and Description Language
KML	Knowledge Modeling Language
KPI	Key Performance Indicator
LEMS	Low Entropy Model Specification
LPNR	Learning Principle of Neuronal Recursion
LRP	Layer-wise Relevance Propagation
LSTM	Long-Short-Term-Memory
LVQ	Learning Vector Quantization
MEA	Modern Enterprise Architecture (3 rd historic category of EA)
ML	Machine Learning
MLP	Multilayer Perceptron
MR	Mixed Reality
MSE	Mean Square Error E_{MS}
MTP	Multi-Target Prediction
NEA	Neuronal Enterprise Architecture
NEM	Neuronal Enterprise Model
NFS	Neuro-Fuzzy System
NKM	Neuronal Knowledge Management
NKMM	Neuronal Knowledge Management Model
NMDL	Neuronal Modeling and Description Language
NN	Neural Network
NPC	Neuronal Process Circle
NPMC	Neuronal Process Modeling Circle
NPOC	Neuronal Process Optimization Circle
NPSC	Neuronal Process Simulation Circle
NS	Neuroscience
NTL	Neuronal Training Language
OL	Organizational Learning
OMG	Object Management Group
OMT	Object Modeling Technique
OOD	Object-Oriented Design
OoI	Object of Investigation
OOSE	Object-Oriented Software Engineering
OR	Operation Research
OSA	Open System Architecture
OSI	Open Source Initiative
P/T net	Place/Transition net (a Petri net variant)
PDCA	Plan-Do-Check-Act cycle or (sometimes Plan-Do-Check-Adjust cycle)

PDSA	Plan-Do-Study-Act cycle
PGE	Product Generation Engineering
PMDL	Process Modeling and Description Language
PMI	Project Management Institute
PML	Process Modeling Language
POP	Process Optimization Philosophy
PROSA	Process-oriented Service Level Agreement
RBF	Radial Basis Function network
RBM	Restricted Boltzmann Machine
RM	Reference Modeling
RNN	Recurrent Neuronal Network
SBML	System Biology Markup Language
SD	System Dynamics
SEKO	Systematic Exhaustion of Knowledge Objects
Seq.	Sequence(s)
SESS	Specific Evaluation and Selection System
SFD	Stock and Flow Diagram (as part of System Dynamics)
SKI	Static Knowledge Identification (as part of SEKO)
SLA	Service Level Agreement
SLL	Single-Loop Learning
SLM	Service Level Management
SLR	Systematic Literature Review
SME	Small and Medium-Sized Enterprise
SML	Simulation Modeling Language
SNKC	Spiral of Neuronal Knowledge Creation
SOM	Self-Organizing Map
SpML	Space Markup Language
SSD	System Structure Diagram (as part of System Dynamics)
ThE	Threshold-Error E_{Th}
TOGAF	The Open Group Architecture Framework
TQM	Total Quality Management
vers.	version
VR	Virtual Reality
XMI	XML Metadata Interchange format
XR	X Reality (any kind of reality, in analogy to the variable x)

Symbols

Set of Neurons

$Bias$	set of neurons in a network as bias neuron
C_j	set of cells within a LSTM block j
G_j	set of neurons within a LSTM block j with a gating function
H	set of neurons in a network's hidden layer
I	set of neurons in a network's input layer
K	set of neurons in a network's output layer
Kl	set of classifications, which contains all combinations of classifications

Number Entities

n_b	number of bias neurons for every biased unit
n_c	number of cells within each LSTM block
n_g	number of gating neurons within a network's LSTM block
n_i	number of input neurons within a network's input layer
n_j	number of LSTM blocks within a network's hidden layer
n_k	number of output neurons within a network's output layer
n_{kl}	number of classification options for Kl

Neuronal Learning

a	activation input of a network's neuron
b	activation output of a network's neuron
δ_k	incoming errors of a network's output neuron k
ϵ_k	outgoing errors of a network's output neuron k

Neuronal Learning

x_i^t	input element of neuron i at time step t
b_k^t	output element of neuron k at time step t (here: network prediction)
t_k^t	target element of neuron k at time step t
α	learning rate, a relativization of weight adjustments during learning
m	momentum, an inertia moment of weight adjustments during learning
∇_{hk}	failure gradient of hidden neurons h and output neurons k , with whom errors are minimized during learning
$\Delta \underline{\mathbf{w}}(n)$	weight adjustment of learning step n , with whom learning is carried out
η	normalization factor, considers e.g. Seq , K and/or T
\hat{x}_i	normalized input element of neuron i
σ_i	standard deviation σ of input elements x_i
m_i	average m of input elements x_i
S	training set
S'	test set
$e(\dots)$	error function, whose result $\underline{\mathbf{E}}^T$ is minimized during learning

LSTM Blocks

$f(\dots)$	transfer function of the gating units of LSTM blocks
$g(\dots)$	transfer function of the cell input of LSTM blocks
$h(\dots)$	transfer function of the cell output of LSTM blocks
$i(\dots)$	transfer function of the output units of a network
l_j	input gate of LSTM block j
ϕ_j	forget gate of LSTM block j
ω_j	output gate of LSTM block j
a_k^t	activation input of a network's output neuron k at time step t
b_k^t	activation output of a network's output neuron k at time step t
$c_j^{n_c}$	cell n_c within the LSTM block j
s_c^t	state of cell c at time step t

Collection of Neuronal Weights

w_{ik}	weights of connections from input units i to output units k
w_{ih}	weights of connections from input units i to hidden units j , as there are LSTM blocks, those include w_{ic} , w_{il} , $w_{i\phi}$, $w_{i\omega}$
w_{ic}	weights of connections from input units i to cells of a LSTM block
w_{il}	weights of connections from input units i to input gates of a LSTM block
$w_{i\phi}$	weights of connections from input units i to forget gates of a LSTM block
$w_{i\omega}$	weights of connections from input units i to output gates of a LSTM block
w_{bh}	weights of connections from bias units b to hidden units j
w_{bk}	weights of connections from bias units b to output units k
$w_{h'h}$	weights of recurrent connections between hidden units h' and h (simple recurrent networks)
w_{hh}	weights of recurrent connections between hidden units j , consisting of internal recurrent connections of the same hidden unit, external recurrent connections of further hidden units and foreign recurrent connections of further network's hidden units, as there are LSTM blocks, those include w_{hc} , w_{hl} , $w_{h\phi}$, $w_{h\omega}$
w_{hk}	weights of connections from hidden units j to output units k as there are LSTM blocks, those include w_{ck}
w_{cg}	weights of connections from cells c of a LSTM block and its gating units g (<i>peephole-connections</i>), those include w_{cl} , $w_{c\phi}$, $w_{c\omega}$
w_{cl}	weights of <i>peephole-connections</i> from cells c of a LSTM block to its input gate
$w_{c\phi}$	weights of <i>peephole-connections</i> from cells c of a LSTM block to its forget gate
$w_{c\omega}$	weights of <i>peephole-connections</i> from cells c of a LSTM block to its output gate

Data Set Construction

n_{InpS}	number of incoming input streams
n_{OutS}	number of outgoing input streams
l_{Seq}	sequence length, number of data elements in one sequence
$Id_{TrStart}$	identifier of the first sequence in sequential training data
Id_{TrEnd}	identifier of the last sequence in sequential training data
$Id_{TeStart}$	identifier of the first sequence in sequential test data

Data Set Construction

Id_{TeEnd}	identifier of the last sequence in sequential test data
l_{Tr}	number of sequences in training data set
l_{Te}	number of sequences in test data set
b_{is}	indicator for isolated training and test data sequences
I_{DS}	source-based information number of data set DS consisting of S and S'

Connection Calculations

$C_{w_{ik}}$	number of connections between input units i and output units k
$C_{w_{ih}}$	number of connections between input units i and hidden layer units h
$C_{w_{cg}}$	number of peephole-connections within LSTM blocks
$C_{w_{bk}}$	number of connections between bias units b and output units k
$C_{w_{bh}}$	number of connections between bias units b and hidden layer units h
$C_{w_{hh}}$	number of recursive connections between units of the hidden layer h
$C_{w_{hk}}$	number of connections between hidden layer units h and output units k

Collection of Neuronal Errors

$\underline{\mathbf{E}}^t$	vector version of the current error (of time step t), contains current error elements E_k^t of all output units k
E^t	scalar version of the current error summed over all output units k , used for the visualization of error mountains
$\underline{\mathbf{E}}^T$	vector version of the training error, contains current training error elements E_k^T of all output units k
E^T	scalar version of the training error summed over all output neurons k , used for the visualization of learning progress
$\underline{\mathbf{E}}^G$	vector version of the generalization error or test error, contains current test error elements E_k^G for all output units k
E^G	scalar version of the test error summed over all output units k , used for the generalization capability of a NN
$\underline{\mathbf{E}}_{seq}^t$	vector version of the current error (of time step t), contains the current error elements $E_{k_{seq}}^t$ of output units k and sequence seq

Model Theory

\mathcal{V}	set of model variables
\mathcal{R}	set of relations affecting a system
p	set of model constants
$f(\dots)$	arbitrary complex function to model a system's evolution
$\mathcal{V}^{(k)}$	discrete version of \mathcal{V} considering time step k
$\mathcal{R}^{(k)}$	discrete version of \mathcal{R} considering time step k
$\mathcal{V}(x, y, z, t)$	set of model variables considering spacial information x, y, z and temporal information t

Original-Model-Mappings

O	original system environment
E	model system environment
sb	system boundary
y	object system
x	model system
$Pre(y)$	pre mapping range
$Po(x)$	post mapping range
$f(\dots)$	bidirectional attribute mapping $f(Pre(y) - Po(x))$
$O \setminus Pre$	pretended attributes
$x \setminus Po$	abundant attributes
c	differentiation constant attributes
C	integration constant attributes

Process Theory

P_{formal}	formal process definition
BP_{formal}	formal business process definition
Z	state space
x_i	state variable with index i
$f(\dots)$	action function, or transfer function
s	initial states

Graph Theory

G	graph G is an ordered pair (V, E) , whose components come from the vertex set V and the edge set E
V	vertices $v \in V$ represent the nodes in the graph G
E	edges $e \in E$ represent the connections from node to node

Selection System

$e_l(\dots)$	human-based evaluation task $e(\dots)$ of test person t_l , expert t_l^+ or non-expert t_l^-
i	index for perspectives P_i
j	index for aspects $M_{i,j}$
k	index for objects of investigation o_k
l^+	index for test persons t_l^+
l^-	index for test persons t_l^-
l	index for test persons t_l
$m_{i,j}$	evaluation aspect m of perspective i with index j
n	total number of elements considered
n_i	total number of perspectives P_i
$n_{i,j}$	total number of evaluation aspects $m_{i,j}$ of perspective P_i
n_k	total number of objects of investigation o_k
n_l	total number of test persons t_l
n_l^+	total number of test persons t_l^+
n_l^-	total number of test persons t_l^-
o_k	object of investigation o with index k
q	index for scale z_q
t_l	test person t with index l
t_l^+	test person t with index l^+
t_l^-	test person t with index l^-
$v_{i,j,k,l}$	valuation of evaluation aspect $m_{i,j}$ of object of investigation o_k by $e_l(\dots)$
z_q	operationalization scale z with index q
D	Cartesian product of P and O , denominated as “domain”
G	Cartesian product of D and Z , denominated as “graph”

Selection System

J	variable focus, refers to a combination of kind of KPI and evaluation base, refers to (μ, A) , $(\mu, +)$, (μ, \pm) , (σ, A) , $(\sigma, +)$ or (σ, \pm)
$M_{i,j}$	set of evaluation aspects $m_{i,j}$
μ	mean over all test persons t_l , perspectives P_i and objects of investigation o_k
O	set of objects of investigation o_k , short: OoI
P	set of evaluation aspects $m_{i,j}$ of all perspectives
P_i	set of evaluation aspects $m_{i,j}$ of perspective i
\mathbf{P}_k	vector of evaluation aspects $m_{i,j,k} \forall (i, j)$ of object of investigation o_k
σ^2	variance over all test persons t_l , perspectives P_i and objects of investigation o_k ,
T	set of test persons t_l , denominated as “all test persons”
T^+	set of test persons t_l^+ , denominated as “experts”
T^-	set of test persons t_l^- , denominated as “non-experts”
R_k	valuation result of object of investigation o_k aggregated over all valuations $V_{i,k}^J$
\mathbf{R}^J	vector of final results R_k of object of investigation o_k in regard with J
$V_{i,k}^J$	aggregated valuation of all evaluation aspects $m_{i,j}$ with index j of object of investigation o_k by test person t_l with $e_l(\dots)$ regarding J
Z	set of scales z_l , denominated as “codomain”
$+$	selection of test persons that refers to knowing persons, denominated as “expert base”
$-$	selection of test persons that refers to non-knowing persons, denominated as “non-expert base”
\pm	selection of test persons that refers to all experts surveyed, denominated as “broad expert base”
\mp	selection of test persons that refers only to the author, denominated as “author base”

ANN System Specification

$\mathcal{R}_J^{(k)}$	outer effects on the model $\mathcal{V}_J^{(k)}$ at time step k and system type J
$\mathcal{V}_J^{(k)}$	the model of system type J at time step k
$f(\dots)$	the function, which transfers $\mathcal{V}_J^{(k)}$ to $\mathcal{V}_J^{(k+1)}$ while considering $\mathcal{R}_J^{(k)}$

ANN System Specification

$\mathcal{R}_{N_{ANN}}^{(k)}$	outer effects on the ANN model $\mathcal{V}_{N_{ANN}}^{(k)}$ at time step k , which refers to the network system type N
$\mathcal{V}_{N_{ANN}}^{(k)}$	the ANN model of the network system type N at time step k
$\mathcal{R}_{P_{ANN}}^{(k)}$	outer effects on the ANN model $\mathcal{V}_{P_{ANN}}^{(k)}$ at time step k , which refers to the process system type P
$\mathcal{V}_{P_{ANN}}^{(k)}$	the ANN model of the process system type P at time step k
$\mathcal{R}_{A_{ANN}}^{(k)}$	outer effects on the ANN model $\mathcal{V}_{A_{ANN}}^{(k)}$ at time step k , which refers to the activity system type A
$\mathcal{V}_{A_{ANN}}^{(k)}$	the ANN model of the activity system type A at time step k
$\mathcal{R}_{NA_{ANN}}^{(k)}$	outer effects on the ANN model $\mathcal{V}_{NA_{ANN}}^{(k)}$ at time step k , which refers to the neuronal activation system type NA
$\mathcal{V}_{NA_{ANN}}^{(k)}$	the ANN model of the neuronal activation system type NA at time step k

ANN System Specification Symbols

a_k	ANN system approximating the activity behavior a with index k of activity systems A
ae_i	data input element ae from activity environment with index i of neuronal activation systems NA
$behCf_p$	inner ANN system relation weight considered as control flow $behCf$ with index p of process systems P
$behOp_m$	ANN system approximating the behavioral operator $behOp$ with index m of process systems P
$bpcCf_q$	inner ANN system relation weight considered as control flow $bpcCf$ with index q of network systems N
$bpcOp_o$	ANN system approximating the business process context operator $bpcOp$ with index o of network systems N
kf_p	inner ANN system relation weight considered as knowledge flow kf with index p of activity systems A
ko_m	knowledge input element ko from process or activation environment with index m of activity systems A
io_o	information input element io from process or activation environment with index o of activity systems A

ANN System Specification Symbols

na_k	ANN system approximating the neuronal activity na with index k of neuronal activation systems NA
ne_i	data input element ne from network environment with index i of process systems P
nkf_p	inner ANN system relation weight considered as neuronal knowledge flow nkf with index p of neuronal activation systems NA
nko_m	neuronal knowledge input element nko from activity or activation environment with index m of neuronal activation systems NA
nio_o	neuronal information input element nio from activity or activation environment with index o of neuronal activation systems NA
pl	ANN system approximating the networking process p with index l of network systems N
pe_i	data input element pe from process environment with index i of activity systems A
r_i	data input element r from reality with index i of network systems N
sa_l	ANN system approximating the sub-behavioral neuronal activity sa with index l of neuronal activation systems NA
sae_j	data input element sae from super activity environment with index j of activity systems A
$sBehOp_o$	ANN system approximating the sub-behavioral operator $sBehOp$ with index o of process systems P
$sBpcOp_p$	ANN system approximating the sub-business-process-context operator $sBpcOp$ with index p of network systems N
se_j	data input element se from simulation environment with index j of network systems N
$snae_j$	data input element $snae$ from super activation environment with index j of neuronal activation systems NA
sp_l	ANN system approximating the sub-process behavior of task sp with index l of process systems P
spe_j	data input element spe from super process environment with index j of process systems P
sn_m	ANN system approximating the non-atomic networking process sn with index m of network systems N
sne_k	data input element sne from super network environment with index k of network systems N
st_l	ANN system approximating the sub-task behavior of task st with index l of activity systems A

ANN System Specification Symbols

t_k ANN system approximating the task behavior t with index k of process systems P

Generic View Specification—Sets

$V_{J_{\mathcal{V}_i}}$	set of inner model vertices of system type J
$V_{J_{\mathcal{V}_o}}$	set of outer model vertices of system type J
$E_{J_{\mathcal{V}_i}}$	set of inner model edges of system type J
$E_{J_{\mathcal{V}_o}}$	set of outer model edges of system type J
$V_{J_{\mathcal{V}}}$	set of model vertices of system type J
$E_{J_{\mathcal{V}}}$	set of model edges of system type J
$V_{J_{\mathcal{R}}}$	set of outer effects vertices of system type J
$V_{J_{\mathcal{V}}}$	set of outer model vertices of system type J
$E_{J_{\mathcal{R}}}$	set of outer effects edges of system type J

Generic View Specification—Coordinates

I	set of original environment systems $i \in I$
J	set of ANN systems $j \in J$, modeling a system from I
K	set of view systems $k \in K$, modeling a system from K
O_i^t	point O as ordered pair $(i_{x_o}^t, i_{y_o}^t, i_{z_o}^t)$ of Euclidean geometry, representing the unique spatial characterization within 3D Coordinate System of the original environment at time step t of a system $i \in I$ and here addressing the x-, y- and z-axis
P_j^t	point P as ordered pair $(j_{x_p}^t, j_{y_p}^t, j_{z_p}^t)$ of Euclidean geometry, representing the unique spatial characterization within 3D Coordinate System of the ANN model environment at time step t of a system $j \in J$ and here addressing the x-, y- and z-axis
Q_k^t	point Q as ordered pair $(k_{x_q}^t, k_{y_q}^t, k_{z_q}^t)$ of Euclidean geometry, representing the unique spatial characterization within 3D Coordinate System of the view model environment at time step t of a system $k \in K$ and here addressing the x-, y- and z-axis
R_k^t	point R as ordered pair $(k_{x_r}^t, k_{y_r}^t)$ of Euclidean geometry, representing the unique spatial characterization within 2D Coordinate System of the view model environment at time step t of a system $k \in K$ and here addressing the x-, and y-axis

Specific View Specification	
$\mathcal{V}_{AOV}^{(k+1)}$	activity overview model characterization AOV at time step $k + 1$ of an ANN system of level A , is set up for any activity object and is based on $f(\mathcal{R}_{AOV}^{(k)}, \mathcal{V}_{AOV}^{(k)})$ corresponding to Eq. 4.51
$\mathcal{V}_{AV}^{(k+1)}$	activity view model characterization AV at time step $k + 1$ of an ANN system of level A , is set up for any activity and is based on $f(\mathcal{R}_{AV}^{(k)}, \mathcal{V}_{AV}^{(k)})$ corresponding to Eq. 4.47
$\mathcal{V}_{FOV}^{(k+1)}$	functional overview model characterization FOV at time step $k + 1$ of an ANN system of level A , is set up for any functional object and is based on $f(\mathcal{R}_{FOV}^{(k)}, \mathcal{V}_{FOV}^{(k)})$ corresponding to Eq. 4.50
$\mathcal{V}_{AandNaIOOV}^{(k+1)}$	information object overview model characterization $IOOV$ at time step $k + 1$ of an ANN system of level A or NA , is set up for any explicit knowledge object and is based on $f(\mathcal{R}_{AandNaIOOV}^{(k)}, \mathcal{V}_{AandNaIOOV}^{(k)})$ corresponding to Eq. 4.69
$\mathcal{V}_{AandNaKGV}^{(k+1)}$	knowledge generation view model characterization KGV at time step $k + 1$ of an ANN system of level A or NA , is set up for any activity or activation and is based on $f(\mathcal{R}_{AandNaKGV}^{(k)}, \mathcal{V}_{AandNaKGV}^{(k)})$ corresponding to Eq. 4.64
$\mathcal{V}_{AandNaKOOV}^{(k+1)}$	knowledge object overview model characterization $KOOV$ at time step $k + 1$ of an ANN system of level A or NA , is set up for any tacit knowledge object and is based on $f(\mathcal{R}_{AandNaKOOV}^{(k)}, \mathcal{V}_{AandNaKOOV}^{(k)})$ corresponding to Eq. 4.68
$\mathcal{V}_{AandNAKOV}^{(k+1)}$	knowledge overview model characterization KOV at time step $k + 1$ of an ANN system of level A or NA , is set up for any knowledge object and is based on $f(\mathcal{R}_{AandNAKOV}^{(k)}, \mathcal{V}_{AandNAKOV}^{(k)})$ corresponding to Eq. 4.67
$\mathcal{V}_{AandNAsV}^{(k+1)}$	set view model characterization SV at time step $k + 1$ of an ANN system of level A or NA , is set up for any data set and is based on $f(\mathcal{R}_{AandNAsV}^{(k)}, \mathcal{V}_{AandNAsV}^{(k)})$ corresponding to Eq. 4.61
$\mathcal{V}_{PBPConPOV}^{(k+1)}$	BP context operator overview model characterization $BPCoPOV$ at time step $k + 1$ of an ANN system of level P , is set up for any BP context operator object and is based on $f(\mathcal{R}_{PBPConPOV}^{(k)}, \mathcal{V}_{PBPConPOV}^{(k)})$ corresponding to Eq. 4.35

Specific View Specification	
$\mathcal{V}_{N_{BPCOV}}^{(k+1)}$	BP context overview model characterization $BPCOV$ at time step $k + 1$ of an ANN system of level N , is set up for any context object and is based on $f(\mathcal{R}_{N_{BPCOV}}^{(k)}, \mathcal{V}_{N_{BPCOV}}^{(k)})$ corresponding to Eq. 4.33
$\mathcal{V}_{N_{NOV}}^{(k+1)}$	network overview model characterization NOV at time step $k + 1$ of an ANN system of level N , is set up for any process network object and is based on $f(\mathcal{R}_{N_{NOV}}^{(k)}, \mathcal{V}_{N_{NOV}}^{(k)})$ corresponding to Eq. 4.34
$\mathcal{V}_{N_{NV}}^{(k+1)}$	network view model characterization NV at time step $k + 1$ of an ANN system of level N , is set up for any process network and is based on $f(\mathcal{R}_{N_{NV}}^{(k)}, \mathcal{V}_{N_{NV}}^{(k)})$ corresponding to Eq. 4.30
$\mathcal{V}_{N_{ScOV}}^{(k+1)}$	scenario overview model characterization $ScOV$ at time step $k + 1$ of an ANN system of level N , is set up for any simulation scenario object and is based on $f(\mathcal{R}_{N_{ScOV}}^{(k)}, \mathcal{V}_{N_{ScOV}}^{(k)})$ corresponding to Eq. 4.27
$\mathcal{V}_{N_{SOV}}^{(k+1)}$	simulation overview model characterization SOV at time step $k + 1$ of an ANN system of level N , is set up for any simulation object and is based on $f(\mathcal{R}_{N_{SOV}}^{(k)}, \mathcal{V}_{N_{SOV}}^{(k)})$ corresponding to Eq. 4.26
$\mathcal{V}_{N_{SV}}^{(k+1)}$	scenario view model characterization SV at time step $k + 1$ of an ANN system of level N , is set up for any process network and is based on $f(\mathcal{R}_{N_{SV}}^{(k)}, \mathcal{V}_{N_{SV}}^{(k)})$ corresponding to Eq. 4.23
$\mathcal{V}_{N_{ACPSOV}}^{(k+1)}$	CPS overview model characterization $CPSOV$ at time step $k + 1$ of an ANN system of level NA , is set up for any CPS object and is based on $f(\mathcal{R}_{N_{ACPSOV}}^{(k)}, \mathcal{V}_{N_{ACPSOV}}^{(k)})$ corresponding to Eq. 4.78
$\mathcal{V}_{N_{ADOV}}^{(k+1)}$	data overview model characterization DOV at time step $k + 1$ of an ANN system of level NA , is set up for any data object and is based on $f(\mathcal{R}_{N_{ADOV}}^{(k)}, \mathcal{V}_{N_{ADOV}}^{(k)})$ corresponding to Eq. 4.90
$\mathcal{V}_{N_{AIOV}}^{(k+1)}$	informational overview model characterization IOV at time step $k + 1$ of an ANN system of level NA , is set up for any informational object and is based on $f(\mathcal{R}_{N_{AIOV}}^{(k)}, \mathcal{V}_{N_{AIOV}}^{(k)})$ corresponding to Eq. 4.85
$\mathcal{V}_{N_{AISOV}}^{(k+1)}$	information system overview model characterization $ISOV$ at time step $k + 1$ of an ANN system of level NA , is set up for any information system object and is based on $f(\mathcal{R}_{N_{AISOV}}^{(k)}, \mathcal{V}_{N_{AISOV}}^{(k)})$ corresponding to Eq. 4.87

Specific View Specification

$\mathcal{V}_{NA_{MOV}}^{(k+1)}$

machine overview model characterization MOV at time step $k + 1$ of an ANN system of level NA , is set up for any machine object and is based on $f(\mathcal{R}_{NA_{MOV}}^{(k)}, \mathcal{V}_{NA_{MOV}}^{(k)})$ corresponding to Eq. 4.86

$\mathcal{V}_{NA_{NAOV}}^{(k+1)}$

neuronal activation overview model characterization $NAOV$ at time step $k + 1$ of an ANN system of level NA , is set up for any activation object and is based on $f(\mathcal{R}_{NA_{NAOV}}^{(k)}, \mathcal{V}_{NA_{NAOV}}^{(k)})$ corresponding to Eq. 4.58

$\mathcal{V}_{NA_{NAV}}^{(k+1)}$

neuronal activation view model characterization NAV at time step $k + 1$ of an ANN system of level NA , is set up for any neuronal activation and is based on $f(\mathcal{R}_{NA_{NAV}}^{(k)}, \mathcal{V}_{NA_{NAV}}^{(k)})$ corresponding to Eq. 4.54

$\mathcal{V}_{NA_{NeOV}}^{(k+1)}$

neuron overview model characterization $NeOV$ at time step $k + 1$ of an ANN system of level NA , is set up for any neuron object and is based on $f(\mathcal{R}_{NA_{NeOV}}^{(k)}, \mathcal{V}_{NA_{NeOV}}^{(k)})$ corresponding to Eq. 4.97

$\mathcal{V}_{NA_{NOV}}^{(k+1)}$

neuronal overview model characterization NOV at time step $k + 1$ of an ANN system of level NA , is set up for any neuronal object and is based on $f(\mathcal{R}_{NA_{NOV}}^{(k)}, \mathcal{V}_{NA_{NOV}}^{(k)})$ corresponding to Eq. 4.96

$\mathcal{V}_{NA_{NTROV}}^{(k+1)}$

non-tangible resource overview model characterization $NTROV$ at time step $k + 1$ of an ANN system of level NA , is set up for any non-tangible resource object and is based on

$f(\mathcal{R}_{NA_{NTROV}}^{(k)}, \mathcal{V}_{NA_{NTROV}}^{(k)})$ corresponding to Eq. 4.89

$\mathcal{V}_{NA_{NV}}^{(k+1)}$

neuron view model characterization NV at time step $k + 1$ of an ANN system of level NA , is set up for any neuronal activation and is based on $f(\mathcal{R}_{NA_{NV}}^{(k)}, \mathcal{V}_{NA_{NV}}^{(k)})$ corresponding to Eq. 4.93

$\mathcal{V}_{NA_{OoOV}}^{(k+1)}$

organizational overview model characterization $OoOV$ at time step $k + 1$ of an ANN system of level NA , is set up for any organizational object and is based on $f(\mathcal{R}_{NA_{OoOV}}^{(k)}, \mathcal{V}_{NA_{OoOV}}^{(k)})$ corresponding to Eq. 4.75

$\mathcal{V}_{NA_{PANNOV}}^{(k+1)}$

process ANN overview model characterization $PANNOV$ at time step $k + 1$ of an ANN system of level NA , is set up for any process ANN object and is based on $f(\mathcal{R}_{NA_{PANNOV}}^{(k)}, \mathcal{V}_{NA_{PANNOV}}^{(k)})$ corresponding to Eq. 4.57

Specific View Specification	
$\mathcal{V}_{NAPOV}^{(k+1)}$	people overview model characterization <i>POV</i> at time step $k + 1$ of an ANN system of level NA , is set up for any people object and is based on $f(\mathcal{R}_{NAPOV}^{(k)}, \mathcal{V}_{NAPOV}^{(k)})$ corresponding to Eq. 4.79
$\mathcal{V}_{NAROV}^{(k+1)}$	role overview model characterization <i>ROV</i> at time step $k + 1$ of an ANN system of level NA , is set up for any role object and is based on $f(\mathcal{R}_{NAROV}^{(k)}, \mathcal{V}_{NAROV}^{(k)})$ corresponding to Eq. 4.77
$\mathcal{V}_{NATROV}^{(k+1)}$	tangible resource overview model characterization <i>TROV</i> at time step $k + 1$ of an ANN system of level NA , is set up for any tangible resource object and is based on $f(\mathcal{R}_{NATROV}^{(k)}, \mathcal{V}_{NATROV}^{(k)})$ corresponding to Eq. 4.88
$\mathcal{V}_{NAUOV}^{(k+1)}$	unit overview model characterization <i>UOV</i> at time step $k + 1$ of an ANN system of level NA , is set up for any unit object and is based on $f(\mathcal{R}_{NAUOV}^{(k)}, \mathcal{V}_{NAUOV}^{(k)})$ corresponding to Eq. 4.76
$\mathcal{V}_{PBehOpOV}^{(k+1)}$	behavioral operator overview model characterization <i>BehOpOV</i> at time step $k + 1$ of an ANN system of level P , is set up for any behavioral operator object and is based on $f(\mathcal{R}_{PBehOpOV}^{(k)}, \mathcal{V}_{PBehOpOV}^{(k)})$ corresponding to Eq. 4.44
$\mathcal{V}_{PBOV}^{(k+1)}$	behavioral overview model characterization <i>BOV</i> at time step $k + 1$ of an ANN system of level P , is set up for any behavioral object and is based on $f(\mathcal{R}_{PBOV}^{(k)}, \mathcal{V}_{PBOV}^{(k)})$ corresponding to Eq. 4.41
$\mathcal{V}_{PIV}^{(k+1)}$	information view model characterization <i>IV</i> at time step $k + 1$ of an ANN system of level P , is set up for any task or sub-process and is based on $f(\mathcal{R}_{PIV}^{(k)}, \mathcal{V}_{PIV}^{(k)})$ corresponding to Eq. 4.82
$\mathcal{V}_{POV}^{(k+1)}$	organization view model characterization <i>OV</i> at time step $k + 1$ of an ANN system of level P , is set up for any task or sub-process and is based on $f(\mathcal{R}_{POV}^{(k)}, \mathcal{V}_{POV}^{(k)})$ corresponding to Eq. 4.72
$\mathcal{V}_{PPOV}^{(k+1)}$	process overview model characterization <i>POV</i> at time step $k + 1$ of an ANN system of level P , is set up for any process object and is based on $f(\mathcal{R}_{PPOV}^{(k)}, \mathcal{V}_{PPOV}^{(k)})$ corresponding to Eq. 4.43
$\mathcal{V}_{PPV}^{(k+1)}$	process view model characterization <i>PV</i> at time step $k + 1$ of an ANN system of level P , is set up for any process and is based on $f(\mathcal{R}_{PPV}^{(k)}, \mathcal{V}_{PPV}^{(k)})$ corresponding to Eq. 4.38

Specific View Specification $\mathcal{V}_{P_{TOV}}^{(k+1)}$

task overview model characterization $T O V$ at time step $k + 1$ of an ANN system of level P , is set up for any task object and is based on

$f(\mathcal{R}_{P_{TOV}}^{(k)}, \mathcal{V}_{P_{TOV}}^{(k)})$ corresponding to Eq. 4.42

List of Definitions

Definition 1	(Model)	21
Definition 2	(Meta-Model)	26
Definition 3	(Process Model)	32
Definition 4	(Business Process Model)	36
Definition 5	(Modeling)	41
Definition 6	(Perspective)	43
Definition 7	(Enterprise Architecture)	47
Definition 8	(Process Modeling Language)	55
Definition 9	(Petri Net)	58
Definition 10	(Petri Net System)	58
Definition 11	(Petri Net Element Characterization)	58
Definition 12	(Transition Rules In C/E Nets)	59
Definition 13	(Knowledge)	100
Definition 14	(Knowledge Process Model)	110
Definition 15	(Knowledge Process Modeling Language)	132
Definition 16	(Simulation Model)	161
Definition 17	(Simulation Modeling)	162
Definition 18	(Simulation Modeling Language)	184
Definition 19	(Learning Organization)	217
Definition 20	(Reference Model)	227
Definition 21	(Fixed Learning Task)	250
Definition 22	(Free Learning Task)	251
Definition 23	(Reinforced Learning Task)	251
Definition 24	(Neuronal Model)	260
Definition 25	(Fuzzy Set)	288
Definition 26	(Fuzzy Systems)	288

Definition 27	(Fuzzy Rule Base)	289
Definition 28	(Fuzzy System)	289
Definition 29	(ANN Modeling Language)	320
Definition 30	(Process Modeling Tools)	341
Definition 31	(ANN Tools)	377
Definition 32	(Neuronal Process Model)	462
Definition 33	(View)	467
Definition 34	(Business Process Context Perspective)	467
Definition 35	(Behavioral Perspective)	469
Definition 36	(Functional Perspective)	470
Definition 37	(Informational Perspective)	472
Definition 38	(Knowledge Perspective)	473
Definition 39	(Organizational Perspective)	474
Definition 40	(Neuronal Perspective)	476
Definition 41	(Process ANN Perspective)	477
Definition 42	(Simulation Perspective)	479
Definition 43	(Object Diagram)	480
Definition 44	(Class)	481

List of Listings

Listing 2.1	NeuroML Description Example with ChannelML	329
Listing 2.2	LEMS-based Description Example with NeuroML vers. 2	333
Listing 2.3	SimPy Simulation Code Example	368
Listing 2.4	MATLAB Deep Learning Toolbox Code Example	370
Listing 2.5	PyBrain Code Example	386
Listing 2.6	Theano Code Example	388
Listing 2.7	Caffe Code Example	391
Listing 2.8	Caffe Code Example Model ‘netModel.prototxt’	392
Listing 2.9	Caffe Code Example Solver ‘netSolver.prototxt’	392
Listing 2.10	TensorFlow Code Example	394
Listing 2.11	Neuron Code Example	402
Listing 2.12	Genesis Code Example	407

List of Figures

Figure 1.1	Placement of the Dissertation within the Intersection of Relevant Topics following Venn's four-set diagram using ellipses (Venn M.A. 1880)	2
Figure 1.2	Research Design of the Dissertation following the Design Science Research Method of Peffers et al. (2007)	9
Figure 1.3	Structure of the Dissertation faced with the Design Science Research Method of Peffers et al. (2007)	11
Figure 2.1	Original-Model-Mappings (synthesized from literature)	24
Figure 2.2	Schemata of Meta-Model System (synthesized from literature)	27
Figure 2.3	Business Processes Environment (synthesized from literature)	33
Figure 2.4	Relation of Mathematical Models and (Business) Processes	37
Figure 2.5	Original-Creator-Model Relationship (synthesized from literature)	38
Figure 2.6	Overview of important types of Petri nets (synthesized from literature)	60
Figure 2.7	Generic Petri net process meta-model (synthesized from literature)	61
Figure 2.8	EPC-based process model example (created to show the strength of the object of investigation)	62
Figure 2.9	House of ARIS (Scheer, 2002)	64

Figure 2.10	Generic ARIS business process meta-model (synthesized from literature)	66
Figure 2.11	EPC-based process model example (created to show the strength of the object of investigation)	67
Figure 2.12	UML diagram types (synthesized from literature)	70
Figure 2.13	Generic UML business process meta-model (synthesized from literature)	71
Figure 2.14	UML-based process model example (created to show the strength of the object of investigation)	73
Figure 2.15	Generic BPMN business process meta-model (synthesized from literature)	77
Figure 2.16	BPMN-based process model example (created to show the strength of the object of investigation)	80
Figure 2.17	Philosophical streams to define knowledge (synthesized from literature)	89
Figure 2.18	Memory and overload theory (synthesized from Baddeley, 1986; Baddeley, Gathercole, and Papagno, 1998; Gathercole, 1998)	93
Figure 2.19	Knowledge Spiral (following Nonaka and Takeuchi, 1995)	109
Figure 2.20	Promote Knowledge Management Model (highlighted elements issued in example)	118
Figure 2.21	GPO-WM Knowledge Management Model (highlighted elements issued in example)	121
Figure 2.22	Potsdam Knowledge Management Model (highlighted elements issued in example)	125
Figure 2.23	Generic Promote business process meta-model (synthesized from literature)	136
Figure 2.24	Promote-based process model example (created to show the strength of the object of investigation)	137
Figure 2.25	Generic GPO-WM business process meta-model (synthesized from literature)	140
Figure 2.26	GPO-WM-based process model example (created to show the strength of the object of investigation)	142
Figure 2.27	Generic KMDL business process meta-model (synthesized from literature)	144
Figure 2.28	KMDL-based process model example (created to show the strength of the object of investigation)	148

Figure 2.29	Portfolio of business informatics methods (following Wilde and Hess, 2006)	155
Figure 2.30	Differentiation of the simulation method (following Thim, 2017, p. 140)	156
Figure 2.31	Abstract schematic of an simulator (following Mildenberger and Sauerbier, 2013)	159
Figure 2.32	Methodology for sequence-oriented simulation studies as overview (synthesized from literature)	163
Figure 2.33	Methodology for simulation studies following Nance (1994)	166
Figure 2.34	Methodology for hierarchy-based simulation studies	167
Figure 2.35	Methodology for prediction-oriented simulation studies following Mildenberger and Sauerbier (1999)	169
Figure 2.36	Simulation taxonomy and CoNM simulation specification (synthesized from literature)	171
Figure 2.37	Generic Space Markup Language meta-model (synthesized from literature)	188
Figure 2.38	Differentiation of flowcharts (created to show the strength of the object of investigation)	191
Figure 2.39	Generic flowchart meta-model (synthesized from literature)	192
Figure 2.40	Flowchart-based simulation model example (created with the aid of AnyLogic to show the strength of the object of investigation, c.f. section 2.6.1.6)	194
Figure 2.41	Generic System Dynamics meta-model (synthesized from literature)	198
Figure 2.42	System Dynamics-based simulation model example (created to show the strength of the object of investigation)	199
Figure 2.43	Generic Business Process Reengineering Meta-Model (synthesized from literature)	222
Figure 2.44	Generic Continuous Improvement Process Meta-Model (synthesized from literature)	225
Figure 2.45	Generic Reference Modeling Meta-Model (synthesized from literature)	228
Figure 2.46	Single-Loop Learning Model for Learning Organizations (synthesized from literature)	233
Figure 2.47	Double-Loop Learning Model for Learning Organizations (synthesized from literature)	235

Figure 2.48	Deutero-Learning Model for Learning Organizations (synthesized from literature)	237
Figure 2.49	Schematic Drawing of Compartments of a Neuron (synthesized from literature and created to show the consistency of neuronal networks)	245
Figure 2.50	A single neuron	261
Figure 2.51	Example of a Multilayer Perceptron	263
Figure 2.52	Example of a Recurrent Neuronal Network	266
Figure 2.53	Mean Square Error E_{MS}^t for $\underline{t}^t = (1.0, 0.0)^T$	274
Figure 2.54	Cross Entropy Error E_{CE}^t for $\underline{t}^t = (1.0, 0.0)^T$	275
Figure 2.55	Threshold Error E_{Th}^t for $\underline{t}^t = (1.0, 0.0)^T$	276
Figure 2.56	Overfitting Compared to Correct Generalization	278
Figure 2.57	Example of a Feature Visualization with GoogLeNet	281
Figure 2.58	Example of Methods Presenting Attribution	284
Figure 2.59	Fuzzy Set Example for the Linguistic Term <i>turn left</i> <i>slightly</i>	287
Figure 2.60	Fuzzy ANN Architecture	291
Figure 2.61	Jordan Network Architecture	298
Figure 2.62	Elman Network Architecture	299
Figure 2.63	Vanishing Gradient Problem	300
Figure 2.64	LSTM Block Architecture Having Two Memory Cells	302
Figure 2.65	Convolutional Neuronal Network Architecture	310
Figure 2.66	Convolutional Layer Architecture	311
Figure 2.67	Pooling Layer Architecture (with a 3x3 Max Pooling Filter)	312
Figure 2.68	Kohonen Network Architecture	314
Figure 2.69	Neuronal Gas Architecture	315
Figure 2.70	Hopfield Network Architecture	316
Figure 2.71	Generic TensorFlow Graph Meta-Model (synthesized from literature)	324
Figure 2.72	TensorFlow Graph Example by TensorBoard (created to show the strength of the object of investigation)	325
Figure 2.73	Generic NeuroML Meta-Model (synthesized from literature)	328
Figure 2.74	NeuroML Example for a Description of the Hodgkin-Huxley Type K ⁺ Conductance Model (created to show the strength of the object of investigation)	329

Figure 2.75	Generic LEMS Meta-Model (synthesized from literature)	332
Figure 2.76	LEMS Example for a Description of the Hodgkin-Huxley Type K ⁺ Conductance Model (created to show the strength of the object of investigation)	333
Figure 2.77	Example of the Simulation Run Realization on Behalf of the ARIS Platform (created to show the strength of the object of investigation)	353
Figure 2.78	Example of ADONIS (created to show the strength of the object of investigation)	357
Figure 2.79	Example of Modelangelo (created to show the strength of the object of investigation)	360
Figure 2.80	Process Simulating with Signavio (created to show the strength of the object of investigation)	363
Figure 2.81	Serpentine Queue Example of AnyLogic (created to show the strength of the object of investigation)	366
Figure 2.82	Code Example for a Simulation with SimPy (created to show the strength of the object of investigation)	368
Figure 2.83	Code Example for a Deep Network with MATLAB (created to show the strength of the object of investigation)	370
Figure 2.84	Example of MATLAB's ANN Construction (created to show the strength of the object of investigation)	371
Figure 2.85	Example of MATLAB's Simulation Design (created to show the strength of the object of investigation)	373
Figure 2.86	Code Example for Solving the Parity Problem with PyBrain (created to show the strength of the object of investigation)	386
Figure 2.87	Code Example for Solving the Parity Problem with Theano (created to show the strength of the object of investigation)	388
Figure 2.88	OpFromGraph Node Example of Theano (created to show the strength of the object of investigation)	389
Figure 2.89	Code Example for Solving the Parity Problem with Caffe (created to show the strength of the object of investigation)	391

Figure 2.90	Configuration Files for solving the Parity Problem with Caffe (created to show the strength of the object of investigation)	392
Figure 2.91	Code Example for Solving the Parity Problem with TensorFlow (created to show the strength of the object of investigation)	394
Figure 2.92	Example of neuroConstruct (chosen to show the strength of the object of investigation, De Schutter and Bower, 1994)	396
Figure 2.93	Example of NeuroTessMesh (GMRV, 2018)	398
Figure 2.94	Code Example for a Soma Simulation using NEURON (created to show the strength of the object of investigation)	401
Figure 2.95	Architecture of GENESIS (Beeman, 2005)	404
Figure 2.96	Genesis Tutorial Example (Beeman, 2005)	406
Figure 2.97	SLR Flow Diagram	417
Figure 3.1	The CoNM Terminology as Binding Part at the ANN Process Domain following Grum and Gronau (2018a)	433
Figure 3.2	Conceptual Reference Frame of the CoNM	435
Figure 3.3	Methodological Approaches for the Realization of a Systematic Literature Review	444
Figure 3.4	Methodological Approach for the Realization of a Systematic Literature Review	449
Figure 3.5	The Design Science Research Methodology Process Model (Peffers et al. 2007)	450
Figure 3.6	Methodological Approach for the Development of Neuronal Networks following Lämmel and Cleve (2012, p. 199)	452
Figure 3.7	Methodological Approach for Empirical Research (Diekmann 2012, p. 186–230)	454
Figure 3.8	Methodological Approach for the Dissertation	458
Figure 4.1	The Design of the Meta-Model Creation	464
Figure 4.2	Iterative Concretion of Perspectives by the Introduction of Central Terms and Definitions	466
Figure 4.3	Object Model Specification	483
Figure 4.4	The Generic Meta-Model	485
Figure 4.5	Systematic Selection Process	487
Figure 4.6	The Design of the Selection System Creation	488
Figure 4.7	The Design of the Selection System	492

Figure 4.8	The Design of the Neuronal Enterprise Architecture Design for the CoNM	497
Figure 4.9	The Design of the Proceeding with the Neuronal Enterprise Architecture	501
Figure 4.10	The Neuronal Process Circles	505
Figure 4.11	The New Components the Neuronal Knowledge Management Model following Grum (2020)	507
Figure 4.12	The Design of the Neuronal Knowledge Management Model	508
Figure 4.13	The Design of the Neuronal Knowledge Management Ordering System (phase-specific)	521
Figure 4.14	Model-ANN-Perspective-View Relations for the CoNM Design (Derived from the CoNM Sub-Artifacts)	523
Figure 4.15	The Generic, Model-Oriented, Graph-Based Business Process Description	538
Figure 4.16	The Interrelation of CoNM Views	540
Figure 4.17	The Omnipresent Modeling Items of the NMDL (Derived From the Meta-Model Designed and Devised For Simple Navigation)	600
Figure 4.18	The Atomic NMDL Taxonomy Views	601
Figure 4.19	The Complex NMDL Taxonomy Views	603
Figure 4.20	The Flow-Oriented NMDL Views	605
Figure 4.21	The Framework for Signal-Dependent Knowledge Detection (Neuron View)	608
Figure 5.1	Common Modeling Modes for the Bidirectional Modeling (following Grum and Gronau, 2018b)	617
Figure 5.2	Common Use Cases for a Bidirectional AR Modeling (following Grum and Gronau, 2018b)	620
Figure 5.3	Architectural Set Up (as UML Deployment Diagram Variant)	622
Figure 5.4	Morphological Box for a Bidirectional AR Modeling (following Grum and Gronau, 2018b)	626
Figure 5.5	Morphological Box for a Bidirectional AR Modeling (following Grum and Gronau, 2018b)	627
Figure 5.6	AR Modeling on behalf of the Augmentor (Example from Grum and Gronau, 2017)	628
Figure 5.7	Workflow of CoNM Modeling (as UML Activity Diagram Variant)	630

Figure 5.8	System Interactions Arranged in Time Sequence (as UML Sequence Diagram Variant)	632
Figure 5.9	Modelangelo Extension—Ground Plan. (Example from Grum and Gronau, 2017)	634
Figure 6.1	The Demonstration of the Selection System, Evaluation of Marcus Grum	645
Figure 6.2	The Demonstration of the Selection System, Evaluation of Experts (Mean)	647
Figure 6.3	The Demonstration of the Selection System, Evaluation of Experts (Variance)	649
Figure 6.4	The Demonstration of the Selection System, Evaluation of all Test Persons (Mean)	651
Figure 6.5	The Demonstration of the Selection System, Evaluation of all Test Persons (Variance)	654
Figure 6.6	The Atomic Neuronal Conversions	658
Figure 6.7	The Pure Complex Neuronal Conversions	659
Figure 6.8	The Abstract Neuronal Conversions	660
Figure 6.9	The Architecture of a Single Neuron (NMDL <i>NeuronView</i>)	661
Figure 6.10	The Architecture of a MLP (NMDL <i>NeuronView</i>)	662
Figure 6.11	The Architecture of a RNN (NMDL <i>NeuronView</i>)	663
Figure 6.12	The Architecture of a LSTM Block (NMDL <i>Neuron</i> <i>View</i>)	664
Figure 6.13	The Training Set for the Live Learning Demonstration (<i>SetView</i>)	667
Figure 6.14	The Knowledge Composition for the Live Learning Demonstration (<i>KnowledgeOverview</i>)	668
Figure 6.15	The Neuronal Composition of the ANN Trained (<i>NeuronalOverview</i>)	669
Figure 6.16	The Training Specification for the Live Learning Demonstration (<i>ActivationView</i>)	669
Figure 6.17	The Live Learning Visualization	671
Figure 6.18	The Training Set for the SKI Demonstration (<i>SetView</i>) ...	676
Figure 6.19	The Knowledge Composition for the SKI Demonstration (<i>KnowledgeOverview</i>)	676
Figure 6.20	The Neuronal Composition of the ANN Trained (<i>NeuronalOverview</i>)	677
Figure 6.21	The Training Specification for the SKI Demonstration (<i>ActivationView</i>)	678

Figure 6.22	The Performance of the ANN Trained	679
Figure 6.23	The SKI Demonstration (<i>NeuronView</i>)	680
Figure 6.24	The SKI Demonstration (<i>KnowledgeGenerationView</i> Variant 1)	684
Figure 6.25	The SKI Demonstration (<i>KnowledgeGenerationView</i> Variant 2)	685
Figure 6.26	The DKI Demonstration (<i>NeuronView</i>)	688
Figure 6.27	The process to be simulated for the NPS Demonstration (<i>ProcessView</i>)	696
Figure 6.28	The Overviews of NPS Example	698
Figure 6.29	The Views of NPS Example	699
Figure 6.30	The Training Specification for the NPS Demonstration (<i>NeuronView</i>)	701
Figure 6.31	The Performance of the ANN Trained	702
Figure 6.32	The Training and Testing Specification for the NPO Demonstration (<i>ActivationView</i>)	706
Figure 6.33	The Initial Network Structure for the NPO Demonstration of <i>NPO Backpropagation</i> Procedures (Experiment 5) and <i>Standard Backpropagation</i> Procedures (Experiment 6)	707
Figure 6.34	The Course of Training and Testing for the NPO Demonstration of <i>NPO Backpropagation</i> Procedures	708
Figure 6.35	The Structural Results for the NPO by <i>NPO</i> <i>Backpropagation</i> Procedures Demonstration (<i>NeuronView</i>)	710
Figure 6.36	The Structural Results for the NPO by Standard <i>Backpropagation</i> Procedures Demonstration (<i>NeuronView</i>)	713
Figure 7.1	Methodological Approach for the Evaluation	720
Figure 7.2	The Evaluation Quality Fields (following Fettke and Loos, 2004a)	721
Figure 7.3	The Evaluation of the CoNM by the CoNM-SESS (Focusing the Mean)	734
Figure 7.4	The Evaluation of the CoNM by the CoNM-SESS (Focusing the Variance)	737
Figure 7.5	Lego Mindstorms EV3 Machines (original)	741
Figure 7.6	The Physical Arrangement of the Lego Simulator Units	742

Figure 7.7	The Ground Plan of the Production Space and Sensor Positioning	743
Figure 7.8	The Lego Simulator Architecture and CoNM Integration (as UML Deployment Diagram Variant)	748
Figure 7.9	The System Interactions Arranged in Time Sequence (as UML Sequence Diagram Variant)	751
Figure 8.1	CoNM Artifacts Realized by this Research	768

List of Tables

Table 2.1	Process and Business Process Comparison	36
Table 2.2	SLR Journal and Conference Selection	414
Table 2.3	Concept Matrix	419
Table 7.1	Requirement Fulfillment	728
Table 7.2	Lego Sensor Design	744
Table 7.3	Lego Scenario Collection	745
Table 7.4	Lego Scenario Probabilities	755
Table 8.1	Design-Science Research Guidelines	776



Introduction

1

1.1 Initial Situation

Being faced by the diverse understanding about the terms *modeling* and *model*, particularly, the case of integrating Artificial Intelligence (AI) and modeling has been widely discussed and the understanding of a model and usage of modeling identified as a research challenge (Fettke 2020) demanding for a complex research agenda (Grum et al. 2020b). Above all, the necessity of comprehensive conceptual discussion of important model characteristics and application possibilities of AI in modeling and vice versa becomes transparent if one accepts the following cases: first, the realization of any kind of process models, simulation models, and ANN models in simulations, and second, the incremental modification of process models to be a process optimization if any form of key performance indicator (KPI) is improved. Up to here, the question by whom and how these modifications or rather simulations are carried out has not been discussed yet. This needs to be considered in conceptual discussions as well. Hence, the thematic placement of the dissertation can be seen at the intersection of four topics, each of which has been visualized with help of an ellipse in Fig. 1.1 and will be considered in the following with regard to the following three: first, the current situation in which this research will be initiated, second, the kind of problem description in this situation, and third, the potential by addressing the problems with the help of a *Concept of Neuronal Modeling* (CoNM).

Supplementary Information The online version contains supplementary material, such as Appendix A, B, C and D whose references are indicated in the text, available at https://doi.org/10.1007/978-3-658-35999-7_1.

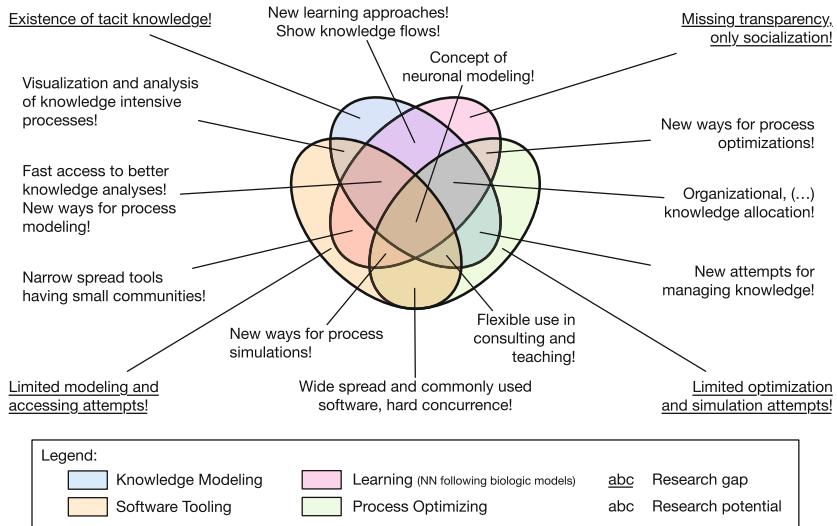


Figure 1.1 Placement of the Dissertation within the Intersection of Relevant Topics following Venn's four-set diagram using ellipses (Venn M.A. 1880)

The Topic of “Learning”: A great potential of neural networks (NN) or artificial neural networks (ANN) in the robust processing of data is well known since nearly four decades. Roughly speaking, these techniques intend to model the capabilities of the (human) brain and copy its working behavior in simulating a network of simple nerve cells. Early ANN architectures go back to the 1940’s; numerous improvements can be found in late 1980 – 2000 (Schmidhuber 2015). Nowadays, in times of big data, enormous amounts of data are available, and the computing power has increased immensely; with this, there exists the possibility to create bigger and more complex networks. Fast graphics processing units (GPUs), which were originally built for fast matrix and vector multiplications in video games, declined in prices and sped up learning in ANN by a factor of 50 and more.

Major contributions to build more complex ANN go to Schmidhuber et al. who created long-short-term-memory (LSTM) blocks, which are much more complex than simple neurons and can deal with the *Fundamental Deep Learning Problem*, which refers to vanishing and exploding gradients (Hochreiter and Schmidhuber 1997b; Gers F., Schmidhuber, and Cummins 2000; and Pérez-Ortiz et al. 2003). Additionally, they claim to be biologically plausible (O'Reilly and Frank 2006).

Major contributions to build bigger ANN go to Le et al. who raised the size of ANN first up to one billion simulated neurons (Le et al. 2011). The researcher named Andrew Ng recently announced to create deep-learning systems even ten times bigger (Jones 2014). This raises their capabilities on the one hand but results in much more complex interpretations on the other. Although various kinds of approaches to interpret ANN are available, the inner working of ANN is mostly seen as black box, which is difficult to interpret. This is because the process of identification of a decision, which was found by the NN or ANN, cannot be explained easily (Tu 1996; Basheer and Hajmeer 2000). Hence, people tend to not trust the ANN performance, and domains such as medicine or banks are not willing to use ANN although they show a great potential for the generation of predictions; for example, the complexity to interpret NN or ANN increases by its size. However, addressing *Concept of Neuronal Modeling* has the potential to describe the inner working of NN and ANN as well as to whiten the black-box of ANN with the help of graphical modeling items.

The Topic of “Knowledge Modeling”: Knowledge can be distinguished in two forms: explicit knowledge, which has been externalized and is impersonal, and tacit knowledge, which is inside people’s heads. The latter cannot be externalized easily (Nonaka and Takeuchi 1995; Gronau, Müller, and Uslar 2004). Nevertheless, it is essential for manifesting a person’s behavior. Following professional literature, there exist thirteen modeling techniques that are discussed and adopted in practices (Gronau 2003; List and Korherr 2006). Hereunder, there are UML-AD, FMC, BML, IDEF, PICTURE, SOM, RAD, LOVEM, EPK eEPK, BPMN, PROMOTE and KMDL. Particularly PROMOTE and KMDL can be identified as inferior candidates for visualizing knowledge (Sultanow et al. 2012) because they anchor knowledge right within the process. Each modeling approach can be extended to operationalize a given knowledge management concept; however, some concepts even bring their proper management approaches. Remus (2002) and Fröming (2009) identified more than 14 knowledge management concepts. The well-established concepts can be found in the following:

- SECI-model and Knowledge Spiral (Nonaka and Takeuchi 1995)
- Model-based Knowledge Management (Allweyer 1998)
- Communication Diagnosis (Dämming, Hess, and Borgmann 2002)
- Business Knowledge Management (Bach, Österle, and Vogler 2000)
- PROMOTE (Hinkelmann, Karagiannis, and Telesko 2002)
- GPO-WM (Heisig 2002) and (Heisig 2005)

- Building Blocks of Knowledge Management
(Probst, Raub, and Romhardt 2006)
- Potsdam Knowledge Management Model, KMDL (Gronau 2009)

Independent of the knowledge of modeling or the management approach selected, knowledge will be involved in specific process steps based on the assumption of consultants and the experience of managers. Since the resulting process model and management effect depend on them, this is highly subjective on the one hand. On the other hand, tacit knowledge cannot be extracted from people's heads so far, and these approaches are limited to non-tacit knowledge items. Here, a technical approach is needed to overcome the subjectivity and limitations mentioned.

Widespread and modern techniques can realize an objective view inside people's heads as follows: Electroencephalography (EEG) shows insights based on the brain's summed electric activation (Niedermeyer and Silva 2004), and magnetoencephalography (MEG) is based on the brain's magnetic activations (Cohen 1972). Functional magnetic resonance imaging (fMRI) shows active regions inside the brain commonly based on the blood-oxygen-level increase (Huettel, Song, and McCarthy 2009). These techniques realize insights in people's heads but are not practical to use within an organizational context: They are expensive, limited by law, hinder everyday work, and do not produce knowledge objects at all; for example, knowledge modeling description languages. Since ANN can be used to approximate human behaviors following biologically plausible models, their working can show tacit knowledge as well. So, the intersection of the first two Venn diagram ellipses is promising. However, the use of ANN-based process models in simulations has not addressed the knowledge generation, yet. Neither simple analyses nor complex ANN-based simulations have focused on tacit or explicit knowledge, although one can expect that the realization of knowledge-intensive process simulations can enable the management of an organization. The *Concept of Neuronal Modeling* so has the potential to address the knowledge generation with the aid of ANN-based simulations and issue the knowledge management of humans as well as ANN-based systems.

The Topic of “Process Optimization”: Within this topic, one can find various approaches to optimize processes. Principally, after the recording and the modeling of existing processes, a systematic identification of weaknesses within the process model shall be realized (Gronau 2016). For this, those process steps are focused upon that produce value for the customer. Here, one can identify three approaches to design and optimize processes: business process reengineering, which issues the redesign of process parts from scratch, continuous improved processes, which issue

the iterative modification of process parts, and reference modeling, which orients process parts to reusable reference models in the sense of best practices (Krallmann, Bobrik, and Levina 2013, p. 230; Gronau 2016; Fettke and Brocke 2019b). Principally, these approaches build on the experience of process optimization experts and their creative thinking about the overcoming of inefficient processes, such that a target process can be modeled (Krallmann, Bobrik, and Levina 2013, p. 243; Gronau 2016). To identify simple process relations and justify the improvement of the process's weaknesses, qualitative tools such as a value chain analysis (Dumas et al. 2013, p. 189), cause-and-effect diagrams (Best 2003, p. 81), etc. can be used. In particular, quantitative tools can be used to drastically reduce subjectivity and argument for or against an optimization measure on behalf of KPIs; examples can be found at various kinds of benchmarks (Nyhuis, Wriggers, and Busse 2008, p. 48), value stream analyses (Engeln and Schwöbel 2005, p. 53), process cost calculations (Heinz et al. 1997; Allweyer 2005, p. 238f; Deuse, Goldscheid, and Finke 2007), return-on-invest-analyses, and process simulations (Gronau 2016).

Regrettably, these kinds of methods and corresponding optimization possibilities are still limited because first, these do not issue quantitative knowledge transfers, second, these tools do not adapt to a given data set on behalf of learning processes, and third, they are limited in regard with their performance accuracy. Since systems that are a part of a process simulations cannot be approximated on an arbitrary detail, they are based on a static design and result in worse performance accuracy. NN and ANN have neither been used for the purpose of optimization of joint process designs nor for the purpose of simulation of joint process designs. Hence, a transfer of ANN to the context of process simulations and optimizations can realize new potentials since they can deal with multi-dimensional, complex relations. Due to the approximation capabilities of ANN, weaknesses of inaccurate simulation results will be addressed and the realism of process models can be evaluated. Further, due to the learning capabilities of ANN, it can be expected to identify inefficient process designs and adapt corresponding designs because of training procedures. Hence, the intersection of the first three Venn diagram ellipses becomes possible because of the construction of a *Concept of Neuronal Modeling*.

The Topic of “Software Tooling”: Within this topic, there can be placed certain modeling, analysis, and simulation tools as well as ANN libraries. On the one hand, there are deep-learning libraries such as Caffe, Theano, TensorFlow, Matconvnet, and LightNet. Focusing on ANN libraries that can deal with LSTM blocks, the following are available: RNNLIB (Graves 2013b), OCropus (Breuel 2015), JANN (Otte, Krechel, and Liwicki 2013), Cortexsys (Cox 2016), PyBrain (Schaul et al. 2010), and Brainstrom (Greff, Srivastava, and Schmidhuber 2015). On the other

hand, there exist various modeling and simulation tools that give Bullinger and Schreiner (2008) as well as Drawehn et al. (2014) as overview. Disregarding the fact of the unavailability of *Concept of Neuronal Modeling*, it is not clear how one of the tools can adapt to this kind of a concept at all. So, the requirements for a tool capable enough to create any form of process models, knowledge models, business process models, simulation models as well as ANN models, to carry out neuronal simulations as well as neuronal optimizations need to be discussed.

Considering those four relevant topics, great potentials become transparent in their intersection, as can be seen in Fig. 1.1. Identifying the placement of the dissertation within the intersection of all relevant topics, the following chapter clarifies the research aims and objectives.

1.2 Research Aims and Objectives

In order to overcome the previously mentioned limitations, this research intends to answer the following main research question:

How can neuronal processes be modeled with help of knowledge modeling description languages?

The following sub-research questions help to structure the main research question and clarify the potential of a modeling of neuronal processes:

1. How can the modeling capability of knowledge modeling description languages be used to raise the transparency of NN?
2. How can the learning capability of NN be used to identify tacit knowledge?
3. How can the modeling of neuronal processes be used for process simulations?
4. How can the learning capability of NN systems be used for process optimizations?

To answer the previously given research questions, the following objectives will have to be realized, each of which is represented by a bullet point and shown next to the corresponding research question:

In the first step, the modeling of neuronal processes has to be defined. The corresponding concept is referred to as “Concept of Neuronal Modeling” (CoNM) from here, and the following objectives will have to be realized.

- A knowledge modeling and description language has to be mapped to NN such that the working of brains can be modeled with the help of a neuronal knowledge modeling description language (NMDL).

- A proceeding is designed that guides the CoNM realization (CoNM Proceeding).
- The creation of an ordering system enables the measure-based evolution of ANN systems and model systems designed. This will be called “Neuronal Knowledge Management Model (NKMM)”.
- Since the amount of new data will be generated in addition to current neuronal proceedings, a systematic approach to exhaust relevant data is needed. This approach will be called “Systematic Exhaustion of Knowledge Objects (SEKO)”.

In the second step, the functioning of the CoNM has to be proven based on a neuronal experiment. Here, focus will be on the identification of tacit knowledge within neuronal activations of a NN that has been trained on a dataset, as sub-question 1 asks for.

- A NN has to be trained based on a dataset; therefore, it is expected to contain tacit knowledge. An analysis has to examine whether tacit knowledge has been identified by the CoNM and whether the NN behavior modeled is plausible (Transparency Experiment).

In the third step, the functioning of the CoNM as a simulator has to be proven based on an experiment. Here, the focus is on the scenario-based use of neuronal activations of a NN, as sub-question 3 asks for.

- A NN has to be simulated based on a dataset so that it can apply tacit knowledge. An analysis has to examine whether the tacit knowledge used, which has been identified by the CoNM before, and the NN behavior modeled is plausible (Neuronal Simulation Experiment).

In the fourth step, the functioning of the CoNM has to be transferred to a process optimization experiment. Here, the focus is on the identification of optimization potentials by NN that stand for real processes, as sub-question 4 asks for.

- A NN has to be optimized based on a dataset that is expected to demand for a different set of tacit knowledge. An analysis has to examine whether the tacit knowledge modification, which has been identified by the step before, and the modified NN behavior modeled are plausible (Neuronal Optimization Experiment).

- CoNM as a learning principle has to be formulated. This is based on the assumption that the architecture of a NN can be interpreted such that it refers to real processes (Learning Principles).

Further, contemporary tools are attractive to be combined, or rather developed, such that the CoNM can be realized easily. Therefore, the following objectives have to be realized:

- A design for the CoNM architecture has to be developed. This will be referred to as neuronal enterprise architecture (NEA).
- A design for the CoNM software has to be developed.
- A design for the CoNM hardware has to be developed.

The outcome of each bullet point can be seen as a sub-artifact that serves as an integral part of the main artifact of the CoNM. As the research questions build on one another, a research design has to consider the dependencies. The following chapter shows an approach for this.

1.3 Proposed Research Design

In order to deal with the previously mentioned dependencies of sub-artifacts, the following chapter clarifies an adequate research design. It follows a design-oriented research approach so that an artifact is realized and aims to overcome an identified problem efficiently. In this research, the main artifact refers to the CoNM that aims to overcome the problem of setting up and using ANN-based process models efficiently.

A design-oriented research aims to achieve the following objectives: The novelty, usefulness, and generality of the realized artifact is clarified to overcome *design objectives* (Hevner et al. 2004). Further, objective descriptions of observations are presented (*description objective*), and in accordance with an *explanation objective*, research insights are explained (Becker 2010). Finally, *proceeding objectives* guarantee the applicability to practical applications such that an evaluation can be carried out (Weber 2015). For this contribution, the realized artifact will stand for those objectives as well.

Although various procedure models are available to carry out design-oriented researches, a universal, established model to carry out these kinds of researches does not exist (Ulrich 2010). The selection of an adequate procedure model and

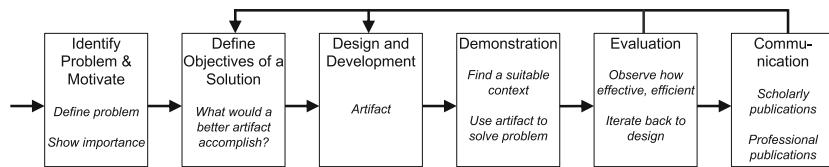


Figure 1.2 Research Design of the Dissertation following the Design Science Research Method of Peffers et al. (2007)

the concrete application to the individual research project has to be realized by the researcher on their own. An overview of design-oriented models is given by Peffers et al. (2006). Building on this overview, Peffers et al. (2007) identified relevant and necessary phases for a design-oriented research so that a more common reference model could be constructed. Since this reference model coherently summarizes phases and tasks for an adequate design-science research, it will help to structure the dissertation's research project as well. The phases can be seen in Fig. 1.2.

Here, one can see cyclic process sequences that can be used to deal with previously mentioned circle dependencies: later generated insights can be considered in the form of adjustments of already built artifacts. However, the process is realized on an artifact level: it is realized one time for the main artifact (CoNM) and further for each sub-artifact as well.

In the context of information system research, Hevner et al. (2004) provided seven guidelines for design science in information system research. These can serve for various purposes during the research project (Zelewski 2007; and Weber 2015). At the beginning of a research, they can support the research orientation and planning of concrete research activities. During the realization of the research project, these guidelines design decisions. At the end of the research, its evaluation, communication, and publication is guided because relevant guidelines can be addressed systematically. The guidelines can be found in the following:

1. **Design as an artifact:** A viable artifact must be produced at a design-science-oriented research. This can have the form of a construct, a method, a model, or its instantiation.
2. **Problem relevance:** Design-science intends to develop technology-based solutions using which relevant business problems shall be addressed.

3. **Design evaluation:** By following well-executed evaluation methods, the quality of the designed artifact as well as practical applicability and economic sense must be rigorously demonstrated.
4. **Research contributions:** The contribution of the designed artifact must be made clear and verifiable in the areas of the artifact, its foundation, and/ or methodology.
5. **Research rigor:** Methods for the construction as well as the evaluation of the designed artifact must be rigorously realized.
6. **Design as a search process:** Considering the laws of the problem environment, any means available can to be used in to reach the desired end.
7. **Communication of research:** The outcomes of design-science research need to be presented for different kinds of audiences such as technology-oriented or management-oriented interested parties.

Hence, when the main artifact of the CoNM has been realized, the critical appraisal of this research is oriented toward those guidelines and will be defined and justified in the concluding section (Tab. 8.1).

1.4 Structure

The following section serves as an overview of the dissertation's structure. It follows the design-oriented research approach of business informatics, as can be seen in Fig. 1.3. A justification of the proceeding is given in chapter 3 after a problem and the objective analysis have been realized and relevant artifacts been identified.

In this figure, typical phases of Peffers et al. (2007) can be seen on the left while the mapping to corresponding chapters is shown on the right. In between, one can find the chapter's content, which will be explained in the following:

Chapter 2 explains the literature overview of relevant topics. On the base of a problem analysis, the research gap is identified.

Chapter 3 defines the dissertation's scope based on the objective analysis and presents methodologies to be realized for the development of the CoNM.

Chapters 4 and 5 show the development of the sub-artifacts needed for the construction of the main artifact in form of the CoNM. In the first step, Chapter 4 develops a specific theory and methodology as a kind of foundation. Building up on this, chapter 5 shows the specific implementation.

Chapter 6 then gives demonstrations of the functioning of the CoNM. Each demonstration focuses on an individual aspect of the CoNM, shows an exemplary way of its operationalization and gives analyses and interpretations of its results.

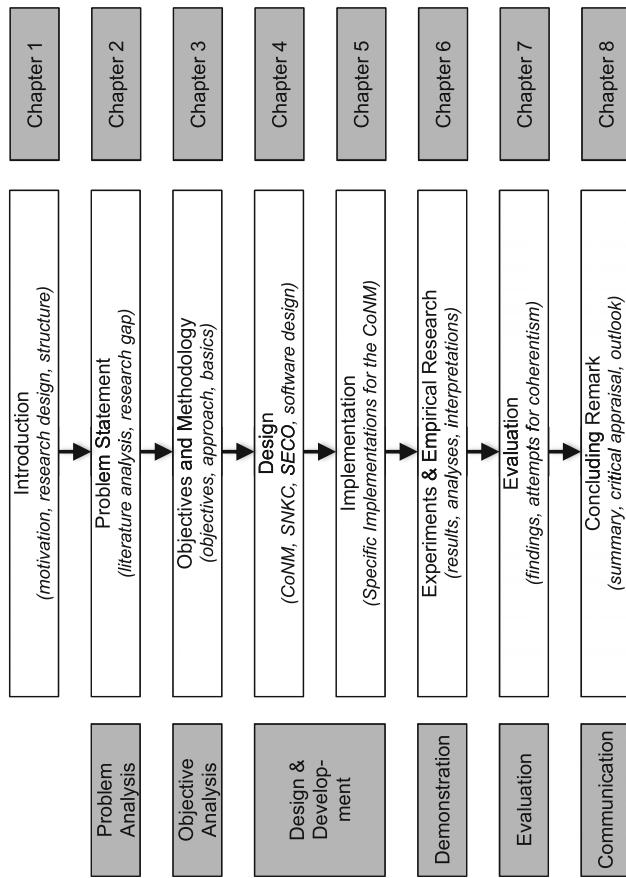


Figure 1.3 Structure of the Dissertation faced with the Design Science Research Method of Peffers et al. (2007)

Hence, the theoretical foundation is enriched by a description of the design knowledge (Fettke, Constantin, and Peter 2010a; Fettke, Constantin, and Peter 2010b). Since each demonstration can be seen as a separate research and development process, it validates the CoNM.

Chapter 7 shows the evaluation of the findings. For this, the findings are shown as a collection and as attempts for coherentism, which are presented w.r.t. the CoNM. Although the artifacts increase their usefulness in combination, each artifact is evaluated separately.

Chapter 8 gives the concluding remark. Based on the summary, the results are considered critically and an outlook is presented.



Problem Analysis and Problem Statement

2

According to Peffers et al. (2007), the problem analysis serves as the first step of a design-oriented research proceeding. In this research, the problem under focus is described and categorized into several aspects. Following Weber, this clarifies selected research questions, supports the development of concrete objectives, and prepares the solution with a foundation of basic concepts (Weber, 2015).

Furthermore, this section is structured as follows: First, the process modeling and knowledge modeling domain is discussed. Thereafter, the simulation and optimization of process models is characterized. Deep learning with neuronal networks is defined after that, following by the issuing of the software tooling of modeling, simulation, and deep learning tools. For each section, the individual problem analysis is presented. Finally, a concluding problem analysis identifies the research gap and builds on a systematic literature review (SLR).

Hereby, each domain-specific section follows the same structure: First, the context of the analysis is established by providing basic concepts and specifying relevant definitions. Subsequently, domain-specific objects of investigation (OoI) are examined. Throughout all analyses, the same analysis criteria have been considered so that different kinds of OoI can be compared irrespective of their nature. Therefore, for instance, modeling tools can be compared with programming libraries or simulation tools and an expectation can be built, provided they are suitable to be a foundation for the CoNM construction. Since analysis criteria and OoI have been identified by the SLR (cf. section 2.7), and can be drawn back, for instance, to the perspectives and modeling objects synthesized in section 4.1, the following presents

Supplementary Information The online version contains supplementary material, such as Appendix A, B, C and D whose references are indicated in the text, available at https://doi.org/10.1007/978-3-658-35999-7_2.

the outcome of iterative analyses. This is in accordance with the methodological approach of this research (compare with section 3.3).

While the quantitative overview across all OoI will be established by an empirical examination in section 6.1 and a ranking will be generated on the basis of the numerical values obtained, the following sub-sections present qualitative explanations about the individual OoI (compare with Fig. 4.5). As the OoI-specific critical appraisals will be presented next to the OoI's characterization with an example clarifying the strength of each OoI, the evaluations can directly be connected with the OoI's characteristics. Nevertheless, explanations are oriented toward the same criteria and categories that will be issued by the selection system in section 6.1. This is to ensure that a comprehensive foundation of the OoI selection and the joint problem analysis can be realized.

2.1 Process Modeling

The process modeling context is characterized by the following: First, basic concepts and definitions are established. Second, different kinds of process taxonomies are provided based on a literature search. Here, relevant perspectives are identified that will serve as the foundation for the creation of the *ANN Process Taxonomy* (section 4.1.1) and stand as a standard for process modeling languages. Third, prominent notations are examined such that they can represent process models. Finally, the domain-specific research need is identified.

2.1.1 Basic Concepts and Definitions

Basic concepts refer to the definition of models (section 2.1.1.1), meta-models (section 2.1.1.2), and business processes as sub-sets of processes (section 2.1.1.3). Finally, the progress of model creation, or modeling, is defined (section 2.1.1.4).

Essentially, the definitions worded out and concepts presented stand as the foundation for following two kinds of investigations: first, for the examination of process taxonomies as objects of investigation as these can be interpreted as building blocks for processes and modeling concepts and consider the management of business processes in organizations, and second, for the examination of process modeling languages (as further OoI) to reflect the processual understanding of organizational activities. Further, the concepts discussed serve as the building blocks for the construction of a CoNM.

2.1.1.1 Models

A literature search of the term “model” yielded various definitions of a multifaceted nature. Due to an inconsistent use of the term in scientific sphere and in everyday life, numerous different interpretations and definition approaches (details can be found at Bernzen, 1990), there is a lack of a consolidated understanding of the concept.

Some authors see the reason for the term’s diversity in the history of its definition (Jammer, 1965, p. 65; Müller, 1983; Bernzen, 1990; Peters, 1998, p. 15; Grum and Gronau, 2018a) because definitions emerge from different schools of thought (Reihlen, 1997, p. 2; Thomas, 2005, p. 8; Grum and Gronau, 2018a). The origin of the term “model” is connected with a dual meaning: the term can either be interpreted as an image of something (e.g. a copy) or an archetype for something (Stachowiak, 1973, p. 129; Thomas, 2005, p. 6).

The authors seek a common conceptual framework to integrate approaches (Bretzke, 1980, p. 28) and establish a common communication and understanding. Since a common framework is not available yet, the following presents definition approaches according to the primary thinking traditions (Grum and Gronau, 2018a). The positioning of original research disciplines is emphasized.

Common Model Theory Definitions: Stachowiak (1973, p. 304) established a common model theory and explicated a formal definition. Based on his analysis of the term, Stachowiak presents three main properties of common models (Stachowiak, 1973, p. 131–133).

The *mapping property* describes models as mappings of an original (e.g. artificial or natural in nature). The mapping refers to a mathematical interpretation of the assignment of an original and model attributes (Stachowiak, 1973, p. 131).

The *truncation property* states that models are truncated by model creators who decide which attributes are relevant. Since the selection of relevant attributes is performed by a subject, the selection cannot be objective, completely determinable, and unique. Hence, different models can describe the same original, and a model can describe various originals.

The *pragmatic property* posits that models are created for a certain subject, for a certain time period, and following a certain purpose. Hence, it is not only relevant *what* a model is about but also *who* and *what* a model is for, and *when* it is current.

Further properties are complemented later (Popp, 1970; Salzmann, 1974; Thalheim, 2010, p. 3120).

In accordance with Stachowiak’s common model theory, a model is defined as the realization on (at least) a quintary predicate relation, which refers to the model x of the original y for the model user k within the period t and the intention z (Stachowiak, 1983, p. 118 and Schütte, 2013, p. 41).

Axiomatic Model Definitions: Based on the formal systems used in mathematics, philosophy, linguistics, and computer science, the definition of an axiomatic model was established within the field of mathematical theory. This was then carried over to the domain of business informatics (Lehner, Hildebrand, and Maier, 1995, p. 28).

Primarily based on the contributions of Tarski (Tarski, 1935; Tarski, 1944; Tarski, 1954; Müller, 1965, p. 154) and focusing on mathematical logic (this includes the set theory, propositional calculus, predicate logic, and so on), a model is considered an interpretation of an axiomatic system, which is established by an individual. Since the syntax of an axiomatic system expressed by a formal language is complemented by the semantic meaning of an individual through its interpretation considering its reality, axioms are either true or false (Hammel, 1999, p. 10; Schulze, 2001, p. 16). Hence, models can be used to examine their *consistency*, *independence* and *completeness* (Bernzen, 1990, p. 425).

While the axiomatic model definition is based on the specialization of a common axiomatic system such that a specific situation is considered, it is completely different from other model definitions that describe models as common abstractions of reality and generalize the specific situation (Klaus et al., 1968, p. 412; Zschocke, 1995, p. 246). Hence, some authors from the fields of economics, computer science, and business informatics separate the “mathematical model” of the field of mathematics (Molière, 1984, p. 40) from a formalized model about an empirical phenomena that uses mathematical symbols such as quantifiers, relations, equations, and inequalities (Mayntz, 1967; Peters, 1998, p. 19). In these approaches, the focus is on the transfer of real problems to mathematical models to optimize the allocation of resources in an algorithmic manner. Examples can be found in the context of investments (Scheer, 1969), distribution planning (Bloech and Ihde, 1972), production planning (Scheer, 1976), and so on.

Mapping-Oriented Model Definitions: Based on contributions from the area of economics, including the decision-making domain (Kosiol, 1961; Schweitzer, 1967; Hax, 1967; Hammann, 1969; Szyperski and Winand, 1974; Bamberg and Coenenberg, 1974; Zentes, 1975; Pohl, 1977), the operations research (Angermann, 1963; Kulhavy, 1963; Kern, 1964; Känel, 1966), and the organization theory (Grochla, 1969), a mapping-oriented model definition was established, which was also accepted in the area of informatics (MacMenamin and Palmer, 1988, p. 36; Hesse et al., 1994, p. 98; Biskup, 1995, p. 39; Goos, 1997, p. 13) and business informatics (Lehner, Hildebrand, and Maier, 1995, p. 27; Becker and Schütte, 1996, p. 19; Heinrich and Roithmayr, 1998, p. 359; Ferstl and Sinz, 2001, p. 18; Alpar et al., 2002, p. 138; Scheer, 2002, p. 18). The mapping-oriented model definition is distinguished from Stachowiak’s common model theory since models are supposed

to have a fixed reference to reality (Thomas, 2005, p. 14). The following definitions clarify this.

According to the approach of Kosiol (1961, p. 321), models are defined as adequate mappings of the considered reality. Moreover, according to Curtis, Kellner, and Over (1992, p. 76), a model is an “abstract representation of reality that excludes much of the world’s infinite detail”. Hence, a model reveals what its creator believes, which does not influence its relevant behavior and thus reduces the complexity for understanding or interacting with a phenomenon. Further, others scholars also agree with this interpretation: Grochla (1969, p. 383) defines a model as the mapping (reproduction) of a system, while Baetge (1974, p. 47) defines models as abstract systems that map (real) systems and simplify details.

Conversely, other scholars emphasize the purpose of a model. While Allweyer and Scheer (1995) view a model as a simplifying, *purpose-related* representation of some part of reality, Becker, Rosemann, and Schütte (1995, p. 435) define a model as an immaterial image of the real world (object system) that follows the purpose of a subject. From this understanding, connections to Stachowiak’s definition can be identified regarding the introduction of a subject and a subject-specific purpose. Rosemann (1996, p. 17) follows a similar definition.

According to Keller and Teufel (1997, p. 117), a model is a simplified representation of reality aiming to make a system’s elements and relationships transparent with the help of a consistent formalization of the system. Furthermore, Whitman et al. define a model as a representation of reality too, which is used for analysis, design, and decision-making (Whitman, Huff, and Presley, 1998), and provides purposes accordingly. Kethers (2000, p. 70) refers to models as abstract representations and complements Whitman’s purposes with *description objectives* and *evaluation objectives*.

The mapping from the original system to the model system can be connected to both a *structural assimilation* and a *content-based assimilation*. Each type of assimilation presents a continuum with which models can be characterized.

The limiting cases of the continuum regarding the structural assimilation either refers to attributes overtaken completely from the object system (*isomorphic* models) or attributes overtaken partly from the object system, which is a kind of reduction, and these models are called *homomorphic* (Schütte, 2013, p. 42). The theoretic possible number of overtaken attributes focusing on structure is zero. In this case, the model system has nothing in common with the original system. The pragmatic minimum number of overtaken attributes focusing on structure is at least one such that the structure-related original model relationship can still be identified.

The limiting cases regarding the content-based assimilation are the following: either models provide the complete content-related nature of the object system

(*isohyl* models) or they only provide parts of the content-related nature of the object system (*analogic* models). In this context, Schütte discusses a semantic recoding or reinterpretation (Schütte, 2013, p. 42). The theoretic possible number of overtaken attributes focusing on content is zero. The model system has nothing in common with the original system. The pragmatic minimum number of overtaken relevant attributes focusing on content is at least one such that the content-related original-model relationship can still be identified.

Construction-Oriented Model Definitions: In contrast to mapping-oriented model definitions, construction-oriented model definitions assume the reality as non-existent and not objectively perceivable as it is subject bound (Thomas, 2005, p. 17). Hence, the creation of a model and the resulting insights are not because of reality mappings but because of the construction of a subject perceiving its individual reality.

Following Rieper (1992, p. 25), problems arise with the construction of models, mostly a perceived discrepancy between achievements and desires. These result from a lack of knowledge on overcoming this perceived discrepancy (Bretzke, 1980, p. 34). Since models are created by subjects using their current knowledge (Watzzlawick, 1984, p. 9), the problems are based on the lack of construction structures (awareness or lack of it) and cannot be simply recognized -problems have to be defined (Bretzke, 1980, p. 35). Hence, the creation of models is connected to a highly creative, iterative construction process, whose construction performance is realized by the model creator.

The construction performance of a subject is widely recognized in the economic literature (Knapp, 1978; Gaitanides, 1979a; Gaitanides, 1979b; Bretzke, 1980; Molière, 1984; Herrmann, 1992), the computer science literature (Ortner, 1997, p. 11; Floyd and Klischewski, 1998), and the business informatics literature (Lechner, 1995; Lechner, 2013; Zelewski, 1995, p. 15; Schütte, 1998, p. 59; Thomas, 2005, p. 17).

The widely accepted definition given by Schütte (1998, p. 59) describes a model as the product of the construction of a model creator declaring the representation of an original for a certain model user and time period using a modeling language (Thomas, 2005, p. 18). Accordingly, it has been common practice since then to define models from the perspective of a construction of process in terms of an engineering task. However, Thomas interprets Stachowiak's definition from a construction-based perspective and defines models as purpose-related representations created through construction processes (Thomas, 2005, p. 25). Here, construction processes refer to sequences of performances belonging together in which drafts (constructions) are created.

System Theoretic Model Definitions: Based on the two assumptions that a system can be separated into its individual components, (1) insights of isolated components can be aggregated (2) and phenomena can be analyzed on a mechanistic level. Analyses on a mechanistic level are well suited for simple systems but not for complex mechanisms such as growth processes, equilibrium processes, meshed control loops, complex decision processes, and so on (Krallmann, Bobrik, and Levina, 2013, p. 45).

Through the creation of a meta-theory, which is called the *general system theory* (Bertalanffy, 1949), Bertalanffy (1972, p. 20) provided a framework to integrate the knowledge of various domains and analyze system components within their context. Hence, network structures can be analyzed, and the framework is well suited for simple and complex mechanisms.

Krallmann, Bobrik, and Levina (2013, p. 53) consider models to be systems that are created through purpose-related abstract mappings of another system. Imboden and Koch (2003, p. 9) also refer to models as simplified representations of complex systems.

While the general system theory was applied in various domains, such as control technology, information theory (Shannon and Weaver, 1949), technical communication (connectionism), chaotic systems, sociological systems (Parsons, 1937; Luhmann, 1984), among others, most definitions focus on open systems influenced by the environment. These definitions enable modeled systems to achieve higher orders and greater levels of heterogeneity that are stable despite environmental fluctuations. Only some definitions focus on closed systems regulating themselves by feedback circles, which is known as cybernetics (Wiener, 1948, p. 20). Relevant application areas are neuronal networks, pattern recognition, groups, or communication (Krallmann, Bobrik, and Levina, 2013, p. 46).

Generally, systems consist system elements and sub-systems, which are somewhat related (Krallmann, Frank, and Gronau, 2002, p. 25). Since the relation can describe the transformation of an input to an output for all process elements and sub-systems, a functional relation becomes transparent (Krallmann, Bobrik, and Levina, 2013). Therefore, the model supports the understanding of and enables interaction with a phenomenon. Therefore, systems can be described by mathematical models with expressions (Imboden and Koch, 2003, p. 9):

$$\{\mathcal{V}\} = f(\{\mathcal{R}\}, \{\mathcal{V}\}, \{p\}). \quad (2.1)$$

Here, $\{\mathcal{V}\}$ represents a set of model variables, $\{\mathcal{R}\}$ a set of relations that affect the system, $\{p\}$ and a set of model constants, and $f(\dots)$ is an arbitrary complex function that models the evolution of a system currently modeled.

According to Imboden and Koch, the following kinds of mathematical models can be found. *Static models* do not consider time-dependent changes (Imboden and Koch, 2003, p. 19) and can be expressed by

$$\mathcal{V} = f(\mathcal{R}). \quad (2.2)$$

Since the state of a system is independent of its previous state, \mathcal{V} is the only dependent variable. Dynamic, time-dependent systems consider the adjustment of system variables with regard to a changed outer relation (disturbance) so that a new equilibrium can be identified (Imboden and Koch, 2003, p. 24). These are modeled as *continuous systems* with

$$\text{temporal change of } \mathcal{V} = f(\mathcal{R}, \mathcal{V}) \quad (2.3)$$

while

$$\frac{d\mathcal{V}}{dt} = \mathcal{V}(t)' = f(\mathcal{R}, \mathcal{V}). \quad (2.4)$$

As system variable changes cannot be characterized by infinitesimally exact values with the aid of the first derivative but the difference of two subsequent time steps, dynamic time-dependent systems can be modeled as *discrete systems* (Imboden and Koch, 2003, p. 26) having

$$\mathcal{V}^{(k+1)} = f(\mathcal{R}^{(k)}, \mathcal{V}^{(k)}). \quad (2.5)$$

Dynamic spacial systems consider time and spatial dimensions. They are modeled with partial differential equations with regard to spatial dimensions x, y, z and temporal dimensions t and are solved numerically with computers (Imboden and Koch, 2003, p. 27, 196), e.g. the following continuous system:

$$\mathcal{V}(x, y, z, t)' = f(\mathcal{R}, \mathcal{V}). \quad (2.6)$$

If systems are not deterministic, a prediction of a system's behavior can be modeled by probability theory and such a system is called a *stochastic system*. This is also required if at least three differential equations are coupled, although their model equations are deterministic (Imboden and Koch, 2003, p. 30). This is called *deterministic chaos* and can be explained by the fact that even nearly identical initial system variables lead to very different system states. Since the measurement of system variables is always connected to errors, absolute similarity is not achievable.

Combining various experiences and methods of different domains, the system analysis offers a practicable method of the general system theory (Fuchs-Wegner,

1972, p. 192). In a heuristic proceeding, initially unknown elements and relations of a system are identified successively through the model creator's observation. Iteratively, irrelevant system elements, sub-systems, and relations are neglected or complemented with regard to a system boundary specified by the model creator. Through the definition of a system boundary, a system environment is identified (Krallmann, Bobrik, and Levina, 2013, p. 44). The influence of the environment on the system can be considered a system input and the influence of the system on the environment can be considered a system output.

Critical Appraisal: Similar to the presented definitions, this contribution intends to center models as abstractions from details without excluding axiomatic reference systems. Since the creation of a CoNM involves the marriage of the neuronal network world with the process modeling world, excluding irrelevant parts of the original system will be a key issue and depends on the model creator.

This contribution agrees with the critique of Bretzke (1980, p. 30–33) with regard to mapping-oriented definition approaches, which interprets the process of a model's construction as a simple reproduction of attributes available within reality. Although the existence of structures within reality can be assumed, their recognition and modeling by the model creator is not guaranteed. Hence, the personal contribution of a model creation is more than the selection of relevant and irrelevant attributes as mapping-oriented definitions imply. It includes the use of available knowledge to create models successively and recognizes the construction performance of a model creator.

Since constructed models support the understanding of and enable interaction with the phenomenon of evolutionary optimized brain functions within the process domain, a model definition might not be limited to the construction of models by subjects. This is the case for all model definition approaches except the general system theory. This enables the integration of several knowledge domains, including mathematical formulated systems using techniques, and semi-formal representations by subjects. This is essential since the first is required for the use of simulations and neuronal techniques and the second is commonly used within the process modeling domain.

Hence, the definition follows a construction-oriented model definition combined with the general system theory, and a model is defined in accordance with Fig. 2.1 as follows:

Definition 1 (Model).

A model is a representation x (denominated as model system \mathcal{V}) of some part of an object system y (e.g. artificial or natural nature), the part being delimited by a

system boundary sb, consists a set of system elements se and sub-systems ss which are connected by a set of relations sr, produce a system output so and abundant attributes a and are affected by outer relations R that include, for e.g., the system input si, all being defined by the interpretation of a model creator mc for the model user mu for description, simulation, analysis, design, and evaluation purposes p within the time period t, being created through construction processes following a consistent formalization f.

Considering this definition, the use of axiomatic models is not forbidden and serves as a promising formalization step going beyond this contribution. Therefore, starting with an empirical phenomenon using mathematical symbols, such as quantifiers, relations, equations, and inequalities (Mayntz, 1967; Peters, 1998, p. 19), the “mathematical model” according to Mollière (1984, p. 40) from the field of mathematics is a promising step.

The use of the term “representation” shall not exclude a mapping relationship of object system and model system rather than including any of the model creator-recognized phenomena represented by any howsoever designed element representation of the original system environment. Faced with abstraction tasks, reduction tasks, enrichment tasks, and so on, this is essential as representations can be found within the human brain.

The use of the term “consistent formalization f” includes, but is not limited to, models expressed by a modeling language. Since models are not stated as results of constructions, drafts of models are also considered. This entails drafts as both explicated or mental models. Hence, even interim results can be expressed with the help of a consistent formalization such as a modeling language extended with mathematical functions. Therefore, such systems can be integrated into a sizeable simulation framework and their progress can be visualized. With this, learning processes, the internal working of a neuronal network (different assimilation types), and the simulation of organizational structures can be characterized over time.

Since organizations are changing constantly, the term “time period t” is considered to reflect model-specific validity periods. This includes periods of the model’s components (sub-systems, relations, and elements) and is essential for ongoing training processes of neuronal networks.

For transparency and reproducibility reasons, the definition integrates circumstances of the model creator and demands a documentation of the model user, model creator, the intended model purpose, the proceeding of the modeling process, and so on. This is essential since the modeling process is subjective and can lead to various valid models, which have to be compared to recognize general and reliable ones.

As a model is defined by a system, it can be interpreted as a collection of entities that act and interact together toward the achievement of some logical end,

which practically realizes a flexible meaning of a system dependent on its objectives (Schmidt and Taylor, 1970; Shishvan and Benndorf, 2017, p. 3).

Model Properties: In the literature, various kinds of model properties can be identified in various definition approaches. Hence, the following presents different model definitions according to the model definition presented here. According to Stachowiak's common model definition, models are characterized by three properties (Krallmann, Frank, and Gronau, 2002, p. 32).

The first property is called the *mapping property*. It entails the question how elements, sub-systems, and relations (from here on referred to as attributes) are mapped from the object system to the model system. This is based on the mathematical (algebraic) definition of sets, and mapping refers to the relation of attributes of the object system and the model system (Stachowiak, 1973, p. 132). According to the mapping-oriented definitions, mapping can be connected to both *structural assimilation* or *content-based assimilation*, and each presents a continuum with which models can be characterized.

The second property is called the *truncation property* and refers to the characteristic that models only providing relevant attributes of an object system neglect irrelevant attributes and underline relevant attributes of the original. These are defined by the perception of the model creator (Stachowiak, 1973, p. 132).

The third property is called the *pragmatic property* and characterizes the reference to a subject. This implies that models fulfill their representation function for a model-using subject, which is applied during a certain time frame and is created for a certain purpose (Stachowiak, 1973, p. 133). Hence, for a complete pragmatic determination of a model, it is not only relevant *what* a model is about, *who* and *what* a model is for, and *when* it is current, as was stated before, but also the following. Thalheim further includes two model properties so that a fourth and fifth property can be found.

The fourth property is called the *extension property* and allows models to represent judgements that cannot be found in the original system (Thalheim, 2010, p. 3120).

The fifth property characterizing models is called the *distortion property* and refers to the improvement of the physical world, the inclusion of visions of a better reality, and so on, which somewhat distorts the observable original system (Thalheim, 2010, p. 3120).

Properties presented here are visualized in accordance with Def. 1 in Fig. 2.1, leading to sets of attributes and three kinds of directed mappings, which are explained below. Here, it becomes clear why mapping directions consider two directions.

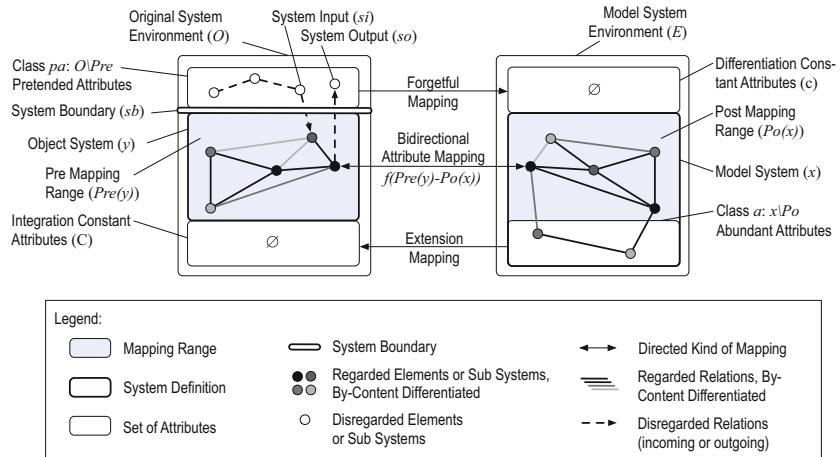


Figure 2.1 Original-Model-Mappings (synthesized from literature)

Attribute Mappings and Sets: When the system border b is specified, attributes of the object system y are truncated from the original system environment O and what is called the *pre mapping range* Pre is provided. This separates the set of the *object system attributes* and the set of the *pretended attributes*. Since attributes of the set of pretended attributes are not carried over to the model system, their relation is called *forgetful mapping* (Thalheim, 2010, p.3120), and the set of *differentiation constant attributes* is empty (based on the mathematical differential calculus definition, constants disappear).

Please note that some of these elements are considered outer relations if they affect the system specified, and some elements of the object system are considered system outputs if they affect the original system environment.

Hence, from the the model creator' s perspective , only relevant attributes are carried over from the *pre-mapping range* to the *post-mapping range*. Since attributes can be found in the original system and model system, their mapping is called *bidirectional mapping* (Schütte, 2013, p. 42), and the arrow in the figure shows arrowheads on both sides because of the following reason: insights and elements can be derived from the original system and transferred to the model system, and insights and elements can be drawn from the model constructed and projected to the original system.

If attributes are provided at the model system, which extends what is known as the *post mapping range Po*, they are denominated as *abundant attributes c* (Stachowiak, 1973, p. 131 as well as Krallmann, Frank, and Gronau, 2002, p. 37). This can refer, for example, to functional input-output relations in computer systems, which is not relevant for the original-model mapping (Stachowiak, 1973, p. 188). Hence, the model x refers to the mathematical model $\{\mathcal{V}\}$ of Eq. 2.1 and consists elements of the post-mapping range and abundant attributes.

Since attributes of the set of abundant attributes can be carried over to the object system but have not been considered because of the lack of knowledge and incomplete specifications, among other reasons (Thalheim, 2010, p. 3129), their relation is called *extension mapping* and the set of *integration constant attributes C* is empty (based on the mathematical integral calculus definition).

Interim Conclusion: These kinds of mappings are relevant for the CoNM to characterize how neuronal networks are assimilated with regard to structure and content (mapping property). This raises the question as to which part of a simulated brain or at which position within the process, for example, knowledge is created. If relevant attributes have been considered by the model creator (bidirectional mapping), overseen by them, and, for example, have been constructed by algorithms (abundant attributes), they can be taken over to the original system as integration constant attributes (extension mapping). Further, the relevance of system attributes can be evaluated with the help of structural assimilation and content-based assimilation.

2.1.1.2 Meta-Models

One speaks from *meta-models* and *meta-meta-models* when models are lifted on a more abstract level so that they can be compared (Kethers, 2000, p. 54). The number of *metas* refers to the level of abstraction and clarifies the kind of models involved in the corresponding meta-model and the meta-models that build on others with regard to abstraction and structure depth (Kottemann and Konsynski, 1984, p. 189). While some meta-model definitions are closely linked to a specific domain, some provide less context-specific definitions.

Domain-Specific Definitions: Jeusfeld and Johnen (1995, p. 237) define meta-models within the data field engineering context as “a language for describing data models”. Nissen (1997, p. 46) defines meta-models within the information system context as a language for the description of information systems, and Kottemann and Konsynski (1984, p. 189) define meta-models within the field of information system development as a computerized implementation of axiomatic models that allow the definition and analysis of median and instantial models. They connect instantial

models to modeled information systems, median models refer to definitions about the system description language, and the axiomatic model is either a set of axioms proving the truth of an assertion or empirical proof through its application over time. Dominguez et al. define a meta-model within the conceptual modeling context as a conceptual schema of objects that are the foundation for a method, data model, or notation (Domínguez, Zapata, and Rubio, 1997, p. 319).

Less Context-Specific Definitions: Hess and Brecht (1996, p. 4) define meta-models as a collection of the most important design objects and their interrelationships. Kethers (2000, p. 55) defines meta-models as the abstract representations of existing or desired models and their interrelationships.

Meta-Meta-Model Definitions: According to Nissen and Jarke (1999), *meta-meta-models* or *m2-models* are models describing and integrating meta-models. Kottemann and Konsynski (1984, p. 188) present the most abstract view of models that can consist various sub-abstraction levels. In compliance with Kethers, the identified levels of abstraction are unlimited and can be specified by the model creator. Consequently, various kinds of modeling environments exist, each connecting two meta-system levels. Examples can be found in the *Usage Environment*, the *Concept Modeling Environment*, and the *Method Engineering Environment*. While the first focuses on the instantiation of models, the second deals with the definition of models for target systems, and the latter provides the definition of modeling languages; for e.g., Schütte (2013, p. 45) supports the idea of an unlimited number of meta-models and speaks of meta-models as models of the n^{th} semantic level that entail models of the $(n - 1)^{th}$ semantic level.

Critical Appraisal: Analogical to the aforementioned definitions, this contribution centers on meta-models as abstractions of models, excluding irrelevant components of models (object system) following a meta-model creator-specific purpose, such that it agrees with Def. 1. Fig. 2.2 reflects basic terms and offers an overview.

Since all meta-models support the understanding and enable the interaction with models, the definition follows the models-about-models philosophy (Schütte, 2013, p. 45), not limiting the number of meta-system levels such that a hierarchy of meta-system levels can be built as required. A meta-model is defined as follows:

Definition 2 (Meta-Model).

A m^n model is a meta-model of the n^{th} level of abstraction representing existing or desired models of the $(n - 1)^{th}$ level of abstraction and their interrelationships.

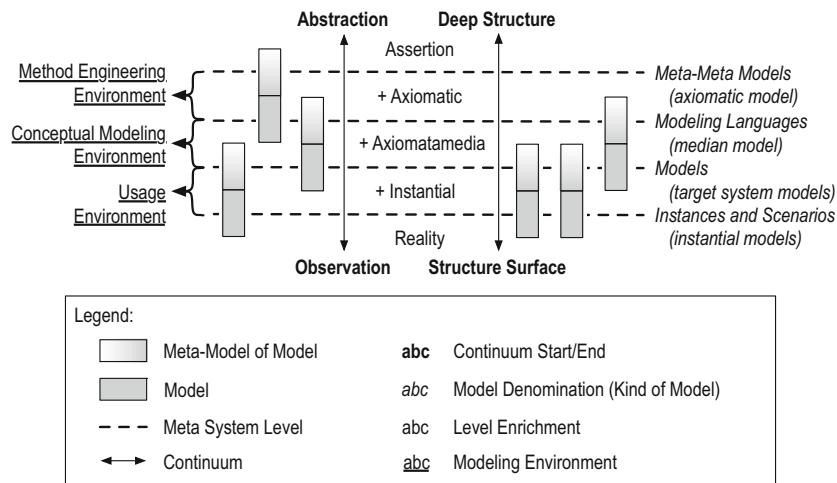


Figure 2.2 Schemata of Meta-Model System (synthesized from literature)

The use of designed meta-models can serve to define *syntax* and *semantics* of models, define modeling languages (Eherer, 1995, p. 16), and describe schema (ISO/IEC-International-Standard, 1990). The use of meta-meta-models therefore can serve to describe and integrate meta-models (Nissen and Jarke, 1999) and compare the capability of modeling languages (List and Korherr, 2006, p. 1533).

Hence, according to this definition, the main artifact of a *Concept of Neuronal Modeling* (CoNM) created here can be designed by the definition of a meta-model about a modeling language for neuronal activities, which can be demonstrated through its application in a modeling language. A meta-meta-model will focus on the integration of meta-models. Hence, this contribution considers the method engineering environment.

2.1.1.3 Processes and Business Processes

A literature search of the terms “process models” and “business process models” yields several descriptions, which mostly present business processes as sub-definitions of processes. Hence, the definition starts with the more general definition of processes and the latter follows. Considering the great and complex history of definitions over centuries (Diamadopoulos, 1960), the following neither demands completeness nor presents detailed concepts but instead presents foundational concepts and justifies their selection.

Processes: The term *process* is commonly associated with three meanings, which come from its usage context (Grau, 2015, p.490): first, a systematical lawsuit (legal proceeding) following common principles in the context of laws; second, the dynamic character of insights, which describe the cognitive advancement as infinite proceedings within the field of knowledge theory; and third, a sequentially structured sequence of successive stages or phases (Rescher, 2006, p.2) that can only be interpreted with regard to a specific philosophical school of thought. Since the third meaning is relevant for this contribution, different schools of thought are evaluated against each other in the following section.

Processes based on a Substantial Being: In compliance with the idea of a *substantial being*, all of nature is pictured as composed of material atoms, and a process refers to the alternation of their positioning in space and time. The origin of this Cartesian interpretation can be found in the atomism of Leucippus, Democritus, and Epicurus (Diels and Kranz, 1906). Since each atom has substantial properties, a process affects only their relation and changes in functionality because of relational changes (Lachmann, 2015, p. 490). Hence, a being is associated with the existence of a substance, and for its being, it only requires itself. Prominent ancient scholars such as Parmenides, Plato, Aristotle, Melissus, and Theophrastus, and versions of substance metaphysics can be found in Hegel, Plotinus, Patrizzi, and Leibniz (Seibt, 2018). Process interpretations based on a substantial being are well-suited for a class of problems that lend themselves to analysis within a framework to which they draw attention (Irvine, 2015).

Processes based on Relational Constituted Being: Following the idea of a *relational constituted being*, a being exists through its relations to other beings and conditioned by the processing of those relations. Here, the being is interpreted in the sense of becoming. Hence, every being is completely or at least partly a component of a process, and the entire world is procedural (Lachmann, 2015, p. 490).

As the very first representative of this school, Heraclitus of Ephesus depicted the world as an interplay of opposing forces, such as a *cosmic fire*, which cause natural processes to change substances (Kirk, Raven, and Schofield, 1983). Accordingly, substances are seen as products of the effects of elemental forces, and material substances are engendered permanently, which is the reason for Heraclitus' famous slogan *panta rhei* ("everything flows"). His fundamental insights were as follows (Seibt, 2018):

1. Processes were assigned with the role as explanatory feature.

So far, processes had been considered as the feature of nature to be explained.

2. Processes form organizational units.
Hence they are measurable and ordered.
3. Processes refer to transitions or alterations.

Further famous scholars are Gottfried Wilhelm Leibniz, Henri Bergson, Charles Sanders Peirce, William James, John Dewey, George Herbert Mead, Alfred North Whitehead, and Charles Hartshorne (Rescher, 2000; Lachs, Lachs, and Talisse, 2008). Only the two most important are considered in the following.

Doctrines of Alfred North Whitehead can be seen as the cornerstone of this school of thought and have been perceived as a codeword for “process philosophy” in recent years (Rescher, 2006, p. 4). Additionally, since he considers the rise and coalescence of entities, processes might be integrative, destructive, or both. This is based on the assumption that nature is composed of events, and objects form an ingredient in the character of events (Whitehead, 1920). Whitehead’s metaphysically primitive notation referred to these events as *actual entities* or *actual occasions*, which are the last real parts consisting reality and can be interpreted as drops of experience, complex and independent (Whitehead, 1929). They comprise the following common structure:

1. Actual entities are constituted by the effects of the previous and by causes related actual entities.
2. A first phase called *prehension* refers to the collection of causal effective actual entities.
3. During a phase known as *concrescence*, a modification and integration of actual entities is carried out. Since this is conditioned by the reproduction of perceived actual entities following teleological perspectives during the prehension, it is not determined and a novelty is introduced, such as creativity.
4. The constitution of actual entities is finalized with the establishment of a common pattern of actual entities based on the concrescence, which is called *satisfaction*.

Hence, becoming real is based on an iterative interplay of the being created in the private and diffused in public. According to Whitehead, this interpretation is superior to the substantial being interpretation because a simple material is senseless, valueless, and purposeless and follows a fixed routine imposed by external relations (Whitehead, 1925). The relation-constituted being interpretation overcomes those issues and considers internal relations as well, which emerge from the nature of the being. Therefore, it is more suited to create a comprehensive and integrated picture of the universe as a whole and is better suited for attempts to understand human psychology and teleology in modern physics (Irvine, 2015).

As a modern representative of the school of thought of a relational-constituted being and as a follower of Whitehead, Nicolas Rescher defines processes to have the following characteristics (Rescher, 2006, p. 2):

1. Processes comprise numerous stages and phases, which are collectively referred to as *complex*.
2. Complexes provide a certain temporal coherence and unity, which means that processes provide an ineliminably temporal dimension.
3. Processes have structures and comprise formal generic forms, such that each concrete process obtains a certain order and form.

Processes based on Speculative Aspirations: Based on Hegel's assumption that the process of reality follows principles explained by philosophical approaches, a collection of *speculative* metaphysics can be identified. Whitehead's metaphor to explain speculative metaphysics describes an airplane flight: while philosophy begins at the ground of the concrete reality of lived experience, the speculation starts with the take-off, which is knowledge creation from imagination. Therefore, facts synthesize into a systematic world-view and must ultimately make contact with the ground (Whitehead, 1929). Although no voice for speculative aspirations can be identified, a growing concern by some philosophers to revive the tradition of speculative metaphysics can be found in papers by Rorty, Lyotard, Foucault, Derrida, Davidson, Dummett, Putnam, Apel, Habermas, Gadamer, Ricoeur, MacIntyre, Blumenberg, and Taylor (Gare, 1999). Important contributions refer to Charles S. Peirce, Samuel Alexander, C. Lloyd Morgan, and Andrew Paul Ushenko (Seibt, 2018). Generally, contemporaries consider them methodologically problematic (Seibt, 2018).

Processes based on Analytic-Interpretative Aspirations: Following the assumption that the description and development of the reality can be characterized without speculations, a collection of *analytic-interpretative* metaphysics can be identified. Here, a processuality is either explained by analyses considering basic categories of common sense and scientific reasoning (analytic philosophy) or characterized by a systematic usage of metaphors following continental philosophy to interpret human experiences and conditions of human existence (Rescher, 2000). Scholars such as William James, John Dewey, Henri Bergson, Martin Heidegger, W.H. Sheldon, and Wilfrid Sellars support this view, all of whom have been abandoned by Western process philosophy (Seibt, 2018).

Processes based on Pragmatics: Interrelated with presented philosophic thinking schools, pragmatic process definitions have been evolved. Its focuses on the natural

and appropriate way to assess a goal-directed practice through its effective and efficient application (Rescher, 2000). Indeed, philosophical and non-philosophical aspirations stand in fruitful symbiosis and the following presents a selection.

Schwickert and Rey (1996, p. 10) defines processes as a chain of logically connected sub-processes that are all directed toward a certain goal being started by a certain event. Kethers (2000, p. 55) considers process models as an abstract representation of a process being constructed for description, qualitative, and quantitative analyses and for design and evaluation purposes.

Critical Appraisal: None of the aforementioned schools of thought and contributions considered a processual interpretation of the reality from the perspective of neurons. Further, the use of technically integrated state-of-the-art deep learning techniques has not been mentioned. Some approaches are better suited as foundation for the creation of a concept of neuronal modeling than others. Therefore, the following critically appraises the presented schools of thought and contributions.

As speculative aspirations deal with abstract concepts, such as the availability of an immortal soul, they cannot be scientifically examined. This school of thought is not suited for the construction of a CoNM. Therefore, the focus shall be on a biological plausibility. Since process definitions based on speculative and analytic-interpretative aspirations have not been accepted by contemporaries and an association with a commonly accepted philosophical opinion is required for the design of a CoNM, such schools of thought are not thus attractive for a definition of processes.

Process definitions based on pragmatics are not attractive in a stand-alone manner since a philosophical base is missing there. Conversely, a philosophical school of thought must not disregard pragmatics so that a design does not remain theoretical but ensures a practical application.

Since the problem of knowledge creation does not belong to a class of problems that lend themselves to analysis within a given framework to which they draw attention—knowledge is created and not substantially available in advance—process interpretations based on a substantial being are not suitable for the construction of a neuronal modeling at all. Hence, for the construction of a neuronal modeling, the consideration of internal relations is required, such as neuronal connections as internal relations of a human's brain. Therefore, knowledge can emerge from the nature of its being and its purpose-related materials, as they are provided by processes based on a relational constituted being (Whitehead, 1925). Since the creation of knowledge coming from an individual entails a highly creative process which is connected to the rise and coalescence of given entities on behalf of those

relations, process definitions based on a relational constituted being are better suited than interpretations based on a substantial being.

Hence, following the intention to create a comprehensive, integrated picture of the universe as a whole from a neuronal perspective, a process definition follows Heraclitus's, Whitehead's, and Rescher's ideas, which ensures a pragmatic connection and, a process model is defined as follows:

Definition 3 (Process Model).

A process model is a model of at least one process or a hierarchy of processes in which each process refers to overlapping, independent, actual entities providing a temporal coherence and unity, which can be measured by the appropriate tools. They are logically from the perspective of an individual as it is understood, altogether form the actual existing world in a multiplicity and become real in the iterative interplay of concrescence in the private and satisfaction in the public domains.

Business Processes: Since business processes are not founded on a process philosophical approach and its corresponding traditional thinking school, the following definitions based on pragmatics focus on a natural and appropriate way to assess the goal-directed business process modeling practice through its effective and efficient application. Hence, business process definitions must be interrelated with philosophical thinking schools presented previously (Rescher, 2000).

According to Tumay (1995, p. 55), business processes can be interpreted as a group of logically connected tasks that use organizational resources. Riekhof (1997, p. 11) specifies business processes as only repetitive, describable, and clearly definable processes. Furthermore, Kethers (2000, p. 56) considers business process models as the abstract representation of a business process constructed for description and qualitative and quantitative analyses, design, and evaluation purposes. Here again, the focus is on the purpose-related use of business process models.

In compliance with Davenport, Hammer and Champy, business processes can be differentiated from ordinary processes by the following five characteristics (Davenport, 1993; Hammer and Champy, 1993):

1. Business processes have customers.
2. Business processes consist of activities.
3. Activities create value for customers.
4. Activities are performed by humans or machines.
5. Business processes can involve several organizational units.

From a system-theoretical perspective, Krallmann, Bobrik, and Levina (2013, p. 220) consider business processes to be activities which create material or immaterial goods in consuming resources whose value is perceived by the customer. Following them, business processes shall only include such activities that create a value for the customer. Others shall be eliminated as they are inefficient (Krcmar and Elgass, 1993, p. 673–695). According to Kueng et al. (1996, p. 101), a customer value focus is not a valid rule for the differentiation of processes and business processes because facts other than business process goals and customers lead to essential activities as well. The generous, purpose-related interpretation of the model creator is essential.

Business Process Models: Since business processes are considered sub-forms of processes, they are part of a company's processual environment. Its so-called *environmental conditions framework conditions* directly influence the execution of business processes (Rohloff, 1995, p. 85). These include *internal factors*, *external factors*, and *location-based factors*, which have to be considered when dealing with business process models (Gronau, 2006b, p. 91). Similar to Schwickert and Fischer (1996, p. 6), the business process environment is visualized in Fig. 2.3 and described in the following.

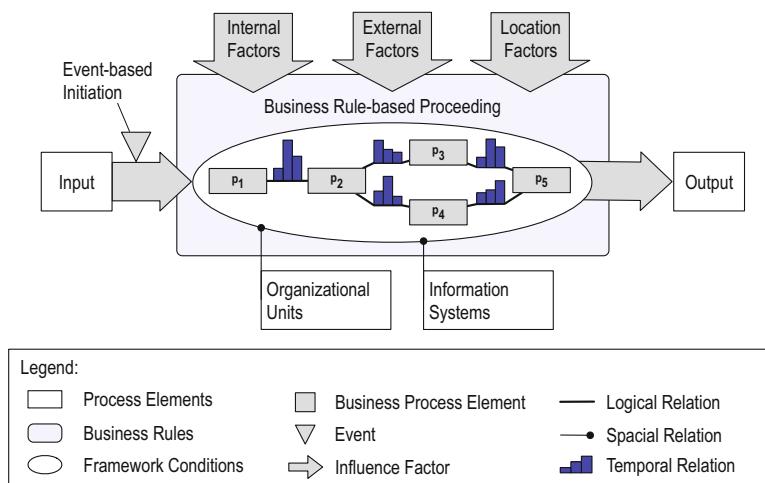


Figure 2.3 Business Processes Environment (synthesized from literature)

The realization of a business process follows the given *business rules* (Knolmayer, Endl, and Pfahrer, 2000, p. 19). Those define the sequence of process elements p_1, \dots, p_5 (Liebelt and Sulzberger, 1992, p. 26–51) and the transformation of material or immaterial input resources to output resources (Peters, 1987, p. 37). Integrating an arising and coalescing of entities refers to the process philosophical perspective on the basis of a relational constituted being (Whitehead, 1920); the valuable construction and destruction are part of the business rule collection as well.

Since internal factors can be oriented toward company objectives and are connected to the appreciation of customers, they can be easily controlled on an operational level. External factors cannot be controlled easily because they are not a part of the company's reach. Hence, external factors, such as legal environmental regulations, economical, technological sociocultural, and ecological environmental factors, are not attractive for short-term process optimization steps (Kreikebaum, 1993, p. 34–40).

Initiated by an event, such as the activation of a time-based indicator that identifies a certain state or the completion of an upstream business process (Klein, 1991, p. 49), a business process ends as its predefined outcome is generated (Krcmar and Elgass, 1993, p. 688).

According to Schwickert and Fischer (1996, p. 8–9), business processes focus on logical, temporal, and spacial relations. Hence, overall, the following relations are relevant for business processes:

- **Logical Relations:** A working order of process steps can be derived by the underlying process logic. It can be identified which activities are independent from each other to be realized in parallel or which activities depend on each other to be carried out sequentially (Schwickert and Fischer, 1996, p. 9).
- **Temporal Relations:** The duration of business processes from the beginning to the end is called *lead time* or *processing time*, which can be divided into many forms such as *process time*, *transfer time*, and *wait time* (Gaitanides, 1994, p. 105). These consider implicitly logical relations.
- **Spacial Relations:** Since process workers and organizational resources, which are consumed or produced within a business process, have a unique position in space, their position within the organizational structure gives rise to further dependencies (Scholz, 1994, p. 49).
- **Transition and Alteration Relations:** A valuable repositioning and alteration of organizational entities is directly linked to a Cartesian interpretation of atomic entities in space and time (Diels and Kranz, 1906).
- **Constructive and Destructive Relations:** The valuable arising and coalescing of entities is directly connected by constructive and destructive actual occasions

(Whitehead, 1920) in business contexts.

Through this, various kinds of process flows can be realized, such as functional flows, performance flows, information flows, organizational flows, objective flows, control flows, resource flows, financial flows, among others, which all can follow opposite directions (Scheer, 2002, p. 11–31).

Critical Appraisal: Generally, definitions consider business processes as sub-definition of processes. The focus on customers and their appreciation is a common element of all definitions. While some approaches prefer the elimination of activities not appreciated by the customer, the CoNM will follow Kueng et al. (1996, p. 101) and their generous, purpose-related interpretation of valuable activities, so that neuronal networks focusing on relevant relations can realize their full potential as the model creator.

A differentiation between processes and business processes is attractive as long as companies can identify processes having high priorities, so that these can be managed efficiently. Some further state business processes can realize a certain use or value for internal or external customers (Hammer and Champy, 1993; Davenport, 1993; Scheer, 1998, p. 3; Gronau, 2017, p. 25) and have interfaces for organization partners, such as customers, suppliers, etc. (Gronau, 2017, p. 25). Considering limited resources, this helps companies focus on important processes. Similar to Schwickert and Fischer (1996, p. 15), Tab. 2.1 presents a comparison.

With respect to both kinds of processes, one can identify a different codification of input objects, events, factors, and output objects (known as codification elements). While business processes are limited to the consideration of codification elements and model them explicitly, ordinary processes consider them as one of many state combinations or action function variations. Further, they differ in their beginning, execution, and end characteristics. While business processes focus on repetitive execution and are finalized as a certain output object is generated, ordinary processes begin once and focus on a permanent execution. The transfer from state to state can be defined commonly for both kinds of formalizations: the repetitive application of a transfer function on current process states, which begins with the initial state, considers interim states and ends with an end state, generating a collection of states that are together called *calculation set C* (Schwickert and Fischer, 1996, p. 3–4). Hence, the transfer function f represents in both cases a mapping of the state space s with the set of possible states within this space (\mathbb{Z}).

A definition of business processes, therefore, is similar to the process differentiation of Davenport (1993), Hammer and Champy (1993), the customer appreciation interpreted by Kueng et al. (1996, p. 101), and the business process environment

Table 2.1 Process and Business Process Comparison

Criteria	Processes	Business Processes
Formalization:	$P_{\text{formal}} = (\mathbf{Z}, f, s)$ $\mathbf{Z} = \{(x_1, \dots, x_n) x_i, n \in \mathbb{N} \text{ and } i = (1, \dots, n)\}$, state variables x_n , action function f and initial states $s \subset \mathbf{Z}$	$BP_{\text{formal}} = (\mathbf{Z}, f, s)$ $\mathbf{Z} = \{\text{input objects, events, factors}\}$, business rules f and initial states $s \subset \mathbf{Z}$
Initial State:	one element out of set of initial states s	one element out of process input
Initiation:	event-based, once	event-based, repetitive
Execution:	permanently	once
State Transfer:	application of an action function f focusing calculation	execution of sub-processes following business rules (focusing relations)
Finalization:	per definition neglected	if output is generated, corresponds to final state
Final State:	one element out of state space \mathbf{Z}	at least one element out of output object set
Task Carrier:	implicit consideration as state or as action function variation	direct assignment
Organizational Resources:	implicit consideration as state or as action function variation	direct assignment

drawn from Schwickert and Fischer (1996, p. 15). Therefore, a business process model is defined as follows:

Definition 4 (Business Process Model).

A business process model is a model of at least one business process or a hierarchy of business processes in which each business process is a sub-form of an ordinary process and is distinguished from processes in terms of the availability of at least one customer who appreciates the created value by the business process execution, by event-based initiation, by business rule-related execution, by the consumption or production of organizational resources, by business process execution by humans or cyber-physical systems, following the intention to create an outcome valued by customers, and by outcome-based finalization.

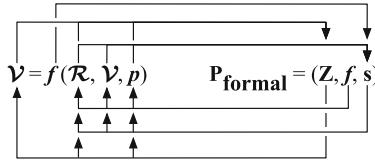


Figure 2.4 Relation of Mathematical Models and (Business) Processes

With respect to this definition, organizations can be interpreted as a system of business processes focusing only on valuable activities (Ferstl and Sinz, 1993, p.589). As this definition does not limit activities to activities appreciated by customers, the potential lies in the detection of valuable activities as well. This potential can only be realized if business processes and side processes, such as environmental changes, non-customer-based developments, nontransparent and non-appreciated processes, etc., are considered jointly. Hence, the creation of a CoNM must consider more than only business processes, and a combination of various model types and their integration to one coherent model will be essential.

As Fig. 2.4 shows, the definition of processes and business processes is in harmony with the model definition of Eq. 2.1.

In this case, the process definition distinguishes between the initial states s and the consecutive states Z , while the mathematical model considers states implicitly in \mathcal{R} , \mathcal{V} , and p . Detailed information about these elements can be found in section 2.1.1.1. Since these are not differentiated by the process definition and business process definition, here, these are considered jointly as states. The following differences result: first, in retrospect, these three cannot be distinguished on the basis of a process state description. Second, the function $f(\dots)$ of the process and business process definition considers the state transfer from s to Z on the basis of more elements than the mathematical model function $f(\dots)$ considers. The latter considers the transfer from the state space $\mathcal{R} \times \mathcal{V} \times p$ to the resulting model V . For example, if the case dynamics are considered, the resulting model refers to the model of the consecutive time step. All together, from a system analysis perspective, parts of processes or business processes can be modeled by mathematical models that represent the realization of generic processes as well as business rules of business processes. Hence, the following considers the formal model representation for the modeling of any kind of processes.

2.1.1.4 Modeling

Generally, the act of the modeling refers to the creation of models independent from the kind of model currently considered. In the literature, one can distinguish between modeling definitions in relation to the context. There exist definitions focusing on common modeling, a meta-modeling, a process modeling, a business process modeling, a simulation modeling, and ANN modeling. Some modeling characteristics can be found in all definitions and must be considered during the establishment of a modeling definition.

Modeling Definitions: According to Schütte (2013, p. 45), modeling starts at the perception of the model creator and an original becomes an original only because of the existence of a model. Perceiving the reality, an internal mental model of the subject's environment is created, which is used for the creation of hypotheses about relations and causations about changes (Krallmann, Bobrik, and Levina, 2013, p. 52). Hence, the perception of the original environment or the model environment and the interaction with them is always based on a highly subjective awareness and the current knowledge of the model creator, which is visualized in Fig. 2.5 in compliance with Krallmann, Bobrik, and Levina (2013, p. 59).

The original environment can be both. It is objective if the reality itself is defined to be the original and it is subjective if a model is defined to be the original (meta-modeling). Since the perception of the original environment is always realized by the model creator, only a subjective reality can be created. This includes integration

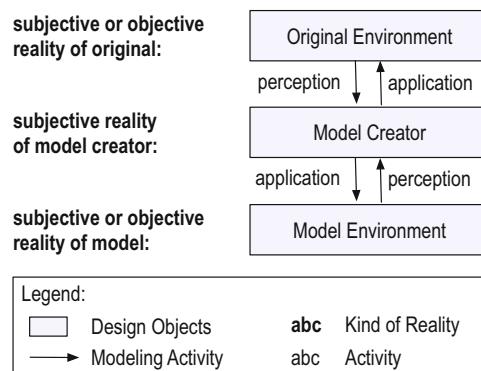


Figure 2.5 Original-Creator-Model Relationship (synthesized from literature)

with mental models of the model creator. If the modeling process does not stop at the subjective reality of the model creator and its subjective reality is applied such that a physical model is created, it can be used within the object reality for model users, for e.g. in simulations, although this model is subjective. If the model creator uses the model on its own, it stays at the subjective reality.

Based on the model environment, insights can be generated, which are subjective interpretations of the model creator within its subjective reality. The subjective insights can be applied in the original environment, which can, for example, be improved. Hence, modeling is a highly creative, arty process, which requires planning, making and applying emphatic judgment skills since numerous viewpoints have to be considered, and judging and balancing simplifications and design decisions (Thalheim, 2012, p. 79).

According to Thomas (2005, p. 25), the creation of models can be interpreted as construction processes because drafts or constructions of the reality are created during this proceeding, which belong to the category of constructions. Thalheim (2010, p. 3116, 3131) supports the interpretation of modeling as a technique and defines the act of modeling to consist the following acts:

- the selection-appropriate modeling language,
- the construction of an appropriate model,
- the application of restrictions,
- the negotiation of conflicting decisions,
- the application of methodologies
- the detection of additional information about the original and improved model,
- the preparation of the model for its use and
- its evolution.

These acts refer to abstract activities, such as *understanding, conceptualizing, abstracting, defining, constructing, refining, and evaluating*, and each activity each is accompanied with the relevant principles (Thalheim, 2010, p. 3117–3126). Since the act of modeling is intended to be an additive process, corrections are possible because of the iterative construction of models (Krallmann, Bobrik, and Levina, 2013, p. 58). Hence, the act of modeling is connected to a repetitive application of various kinds of principles.

Further, modeling can be interpreted as a technological process enabling the model creator to systematically apply knowledge to introduce technological innovations into the planning and development stages of a system (Thalheim, 2012, p. 78). Here, the focus lies on the fabrication and use of artifacts.

Generally, definitions of modeling are connected to the following three activities: First, the application of modeling language constructs, which includes the understanding of syntaxics, semantics, and pragmatics. Second, the application domain gathering, which includes the understanding of problems to be solved, their solution design, requirement specification, and architecture design. Third, the activity of engineering, which includes the encapsulation of experiences (Thalheim, 2012, p. 77).

According to Shishvan and Benndorf (2017, p. 3), the act of modeling goes along with a simplification so that it refers to the process of producing a simplified representation of a complex system of interest. While the degree of simplification results in a different understanding about models in simulation contexts, it differentiates simulations and emulations (issued in section 2.3.1.1); a corresponding differentiation is not present beside simulation contexts.

Modeling Characteristics: Following Curtis, Kellner, and Over (1992, p. 78), the act of modeling can be *descriptive*, *prescriptive*, or *proscriptive*. While the first focuses on the description and understanding of models in place as well as related aspects (Briand et al., 1998, p. 5), the second implies that models such as process models should be used or performed in a certain way (Curtis, Kellner, and Over, 1992). The latter focuses on behavior that is not allowed (Curtis, Kellner, and Over, 1992).

According to Krallmann, Bobrik, and Levina (2013), the act of modeling follows a certain *kind of proceeding*, which is either *top-down*, *bottom-up*, or *hybrid*. While the first refines systems successively beginning with the whole (system and environment) and ending at the intended level of detail (inclusive a hierarchy of sub-systems, elements and relations), the second starts from the intended level of detail and ends at the whole. Although the second kind of proceeding guarantees technical feasibility, an integration is not guaranteed as is with the first approach. As a compromise, hybrid kinds of proceeding combine both top-down and bottom-up approaches (Krallmann, Bobrik, and Levina, 2013, p. 60). Following the intention to reduce the complexity of the act of modeling, modeling languages often provide a proper *modeling method*, which structures acts of modeling (Gronau, 2006b, p. 102).

Meta-Modeling Definitions: When focusing on the creation of meta-models, one speaks form *meta-modeling*. Following Nissen and Jarke (1999), meta-modeling is associated with the creation of models about models. This includes the collection of design objects (Hess and Brecht, 1996, p. 4) and the implementation of axioms (Kottemann and Konsynski, 1984, p. 188). So, for example the formalization of a

modeling language can be specified and with the specification, it can be controlled how process models and their instances look like.

According to Kethers (2000, p.54), meta-modeling is further associated with the activity of lifting models to a more abstract (meta) level, such that they can be compared and integrated. This is ideal for the synthesis of several modeling languages.

Since the number of hierarchical meta-levels is not limited (Schütte, 2013, p.45), an arbitrary generic model about modeling languages can be defined. In accordance with Fig. 2.2, thus, a meta-model can be set up on the basis of instantial models, target system models, median models as well as axiomatic models.

(Business) Process Modeling Definitions: Originally, the term *process modeling* comes from the context of software process modeling (Kellner and Hansen, 1988). Mayer and DeWitte (1999) define process modeling as the mechanism for the construction of a simplified or ideal view of a process, which serves for quantitative analyses. In contrast, the term *business process modeling* refers to the construction of a simplified or ideal view of a business process, which serves for quantitative analyses.

Critical Appraisal: Analogical to the aforementioned definitions, this contribution centers on definitions considering modeling as a technique based on principal-based acts of modeling being structured by modeling methods executed by a subjective model creator, and a modeling is defined as follows:

Definition 5 (Modeling).

Modeling refers to the act of the perception of the model creator's reality and the use of the model creator's knowledge for applying a modeling language, understanding the application domain, and realizing engineering activities in a repetitive, systematic, and principal-based manner, such that a model is constructed within a subjective or objective reality.

Since this definition is not specific about the type of models being created, it includes the construction of all types, for e.g. common models, meta-models, process models, simulation models and business process models. Hence, the inclusion of the type of model being created leads to a specification of the definition presented here.

While the construction of the CoNM requires meta-modeling, because relevant design objects are collected and different modeling language can be integrated, the use of neuronal techniques focuses on the construction of business process

models. Following Thalheim's definition, the construction of a CoNM must involve a modeling language and a method for the application and tool support. These will be handled as individual artifacts to be designed.

2.1.2 Process Taxonomies

The classification of process model terms by certain criteria, categories, or classes such that they are integrated in one common system is referred to as process taxonomy (Bloom et al., 1956; Giaglis, 2001). In the context of business process modeling, several taxonomies have been suggested. In principle, taxonomies can be classified as perspective-oriented, approach-oriented, or architecture-based taxonomies, and the following discusses these. While the first two kinds of taxonomies are suitable for inspections on a meta-level of process models, such as the availability of neuronal perspectives and neuronal modeling objects, the latter is suited for inspections about the use of neuronal techniques.

2.1.2.1 Perspective-Oriented Taxonomies

Perspective-oriented taxonomies categorize modeling approaches on behalf of *perspectives*. Various definitions of the term perspective can be found in the literature. Following the perspective of Nissen (1997, p. 21), the model of an existing or virtual problem is considered conceptually from the viewpoint of one or more stakeholders. Since ANN shall be used to approximate the behavior of various process participants, different perspectives are required to capture the interaction of different stakeholders and their personal notation of knowledge.

According to Becker, Ehlers, and Schütte (1998), perspectives refer to views on a problem area being defined by a certain purpose and the corresponding consequences. Since the CoNM intends to realize optimizations regarding a specified optimization dimensions—this specifies the purpose dimension—and carries out changes within as-is processes. Furthermore, perspectives are required to capture consequences of purpose-related changes (consequence dimension).

Kethers (2000, p. 66) defines perspectives to capture the conceptual model of an existing or virtual problem area from the perspective of one or more stakeholders, in a certain notation, and for a certain purpose. Although this definition combines Nissen's and Becker's definitions, the viewpoint of artificial neuronal instances is not considered here. Hence, this contribution refrains from using the term *view* as synonym for *perspective* from hereon.

Definition 6 (Perspective).

A perspective captures the conceptual model of an existing or virtual problem area from the viewpoint of one or more stakeholders or ANN instances in a certain notation and for a certain purpose.

The following perspectives can be found in the literature:

Functional perspective: The process elements implemented including the flow of informational entities (e.g. data, artifacts, products) that are relevant for the performance of those process elements are represented in the functional perspective. Representatives are Curtis, Kellner, and Over (1992, p. 77), Sowa and Zachman (1992), Keller and Teufel (1997), Vernadat (1996), Vernadat (1998), Whitman, Huff, and Presley (1998), Giaglis (2001), Scheer (2002), Specker (2005, p. 35), Gronau (2012), Grum and Gronau (2017), as well as Grum and Gronau (2018b). According to Whitman et al., the functional perspective can be visualized by an activity view.

Activity-oriented perspective: The process elements implemented exclusive of the flow of entities are visualized within the activity-oriented perspective. Here, the focus lies on the sequence of process tasks and activities performed by process participants. Representatives are Olle, Sol, and MacDonald (1991), Kethers (2000, p. 67), Jarke and Kethers (1999), Specker (2005, p. 35), Gronau (2012), Grum and Gronau (2017), as well as Grum and Gronau (2018b).

Behavioral perspective: When process elements are implemented and aspects of their performance are represented in the behavioral perspective. This draws attention to sequential, parallel, and iterative kinds of processing. Hence, feedback-loops and complex decision-making can be visualized considering input and output criteria. Representatives are Olle, Sol, and MacDonald (1991), Curtis, Kellner, and Over (1992, p. 77), Sowa and Zachman (1992), Frank (1993), Keller and Teufel (1997), Vernadat (1998), Whitman, Huff, and Presley (1998), Giaglis (2001), Specker (2005, p. 35), Gronau (2012), Grum and Gronau (2017), as well as Grum and Gronau (2018b). A business process view was introduced by Whitman et al. to visualize the behavioral perspective. Scheer (2002) combines available perspectives in a behavioral view called control view.

Organizational perspective: Where and by whom in the organization the process elements are implemented is represented in the organizational perspective. This includes the physical location of entities, physical transfer of entities, the storage of entities, etc. Further, this includes organizational units, roles, individuals, human,

and automatic resources (Sultanow et al., 2012, p. 3310). Representatives are Curtis, Kellner, and Over (1992, p. 77), Sowa and Zachman (1992), Frank (1993), Vernadat (1996), Keller and Teufel (1997), Vernadat (1998), Whitman, Huff, and Presley (1998), Giaglis (2001), Scheer (2002), Specker (2005, p. 35), Frank (2014), as well as Grum and Gronau (2018b).

Informational perspective: The informational entities produced or manipulated during a process performance is visualized in the informational perspective. This includes both the structure and relationship of informational entities. These refer to dates, artifacts, and products (tangible, non-tangible). Representatives are Olle, Sol, and MacDonald (1991), Sowa and Zachman (1992), Curtis, Kellner, and Over (1992, p. 77), Frank (1993), Vernadat (1996), Keller and Teufel (1997), Vernadat (1998), Whitman, Huff, and Presley (1998), Giaglis (2001), Scheer (2002), Frank (2014), Grum and Gronau (2017), as well as Grum and Gronau (2018b). A business rule or information view was introduced by Whitman et al. to visualize the informational perspective.

Information flow-oriented perspective: The informational entities produced or manipulated during a process performance and their flow among process participants is visualized in the information flow-oriented perspective. Representatives are Nissen (1997), Jarke and Kethers (1999), Kethers (2000, p. 67), Gronau (2012), Grum and Gronau (2017), as well as Grum and Gronau (2018b).

Resource-oriented perspective: The kind of resources a company manages and the kind of capabilities the resources have is shown in the resource-oriented view in accordance to Vernadat (1996), Whitman, Huff, and Presley (1998), as well as Grum and Gronau (2018b). Often, this is subsumed by the informational perspective.

Knowledge and Knowledge flow-oriented perspective: The kind of knowledge that exists within an organization and the flow of knowledge in and between organizational entities is visualized in knowledge-oriented perspectives. Representatives are Remus (2002), Fröming (2009), Gronau (2012), Sultanow et al. (2012, p. 3310), Grum and Gronau (2017), as well as Grum and Gronau (2018b).

Business process context perspective: The manner in which activities are embedded in the business context is represented in the business process-oriented perspective. This includes goals, actions, measures, and process types. Representatives are List and Korherr (2006, p. 1533), Sultanow et al. (2012, p. 3310), as well as Grum and Gronau (2018b).

Strategic perspective: The kind of goals a process participant has and the interdependencies of goals are visualized in the strategic perspective. Representatives are Sowa and Zachman (1992), Kethers (2000, p. 67), Jarke and Kethers (1999), Frank (2014), as well as Grum and Gronau (2018b).

Service-oriented perspective: The kind of services used from the view point of the customers and suppliers of each process and subprocess is visualized in the service-oriented perspective and can be aggregated within the strategic perspective. Here, communication processes focus on four kinds of phases, which are request, commitment, performance, and evaluation. Representatives are Kethers (2000, p. 67), Jarke and Kethers (1999), Scheer (2002), Grum and Gronau (2018b). According to Jarke and Kethers (1999), speech-act oriented perspectives can be included in service-oriented perspectives.

Communication-oriented perspective: The process instances that communicate with each other are visualized within communication-oriented perspectives (Remus, 2002; Fröming, 2009; Gronau 2012; Grum and Gronau, 2017). This includes the planning level of a communication, its frequency, position of communicators and the time point at which a communication channel is activated (Gronau 2012, p. 106, 108, 136, 137).

Simulation perspective: The manner in which process models can be tested in scenario-based simulations is visualized in the simulation perspective. This further characterizes how scenarios are initialized, which data is produced by simulation instances, and how it is connected to organizational objectives (Grum and Gronau, 2018b).

Interim Conclusion: None of the perspectives presented here consider an integration with neuronal techniques as follows:

- Perspectives do not consider neuronal modeling objects or stand for ANN perspectives.
- These perspectives, therefore, do not consider an integration of ANN perspectives with common modeling perspectives.
- None of the provided perspectives are connected with neuronal libraries, neuronal tools, neuronal methods, or neuronal concepts. Because of this limitation neither models can be executed in terms of a neuronal application nor the behavior of neuronal simulations can be captured by the perspectives.

2.1.2.2 Approach-Oriented Taxonomies

Approach-oriented taxonomies categorize modeling approaches on behalf of the following specializations:

Activity-oriented approaches: Approaches that focus on activities (sometimes referred to as tasks) and only consider information flows, organizational units, data, and so on with respect to activities are referred to as activity-oriented approaches. Authors who have discussed the same include Kueng et al. (1996), Kethers (2000, p. 67), Remus (2002), Fröming (2009), Gronau 2012, as well as Grum and Gronau (2017).

Role-oriented approaches: Approaches that focus on activities from the perspective of a defined role concept are referred to as role-oriented approaches. Considering role-activity associations, responsibilities become clear and roles describe synchronization aspects between activities. Representatives of this concept include Kueng et al. (1996), Kethers (2000, p. 67), Remus (2002), as well as Fröming (2009).

Object-oriented approaches: Approaches developed on object orientation principles of the software design and implementation are referred to as object-oriented approaches. Under these principles, one can find the encapsulation, specialization, generalization, and so on (Kueng et al., 1996; Kethers, 2000, p.67).

Speech-act-oriented approaches: Approaches that focus on communication processes involving a customer, supplier, and performer are referred to as speech-act-oriented approaches. Typically, these processes comprise the sequential phases *request*, *commitment*, *performance*, and *evaluation*. Representatives are Kueng et al. (1996) and Kethers (2000, p. 67).

Interim Conclusion: None of the approaches presented here consider an integration with neuronal techniques as follows:

- Models created do not consider neuronal modeling objects or stand for ANN perspectives.
- Approaches therefore do not consider an integration of ANN perspectives with common modeling perspectives.
- None of the provided approaches are connected with neuronal libraries, neuronal tools, neuronal methods, or neuronal concepts.

2.1.2.3 Architecture-Based Taxonomies

Only some approaches go beyond the design of process models and additionally consider the design of information systems required for the creation, implementation, use, and optimization of designed (business) process models. As information system components can be derived from and are not limited to modeling perspectives (see section 2.1.2.1) or modeling approaches (see section 2.1.2.2) for underlying information systems, the approaches realize proper architectures and belong to architecture-based taxonomies. Referring to the first formal definition of enterprise architectures (EAs), they are defined as follows (Richardson, Jackson, and Dickson, 1990, p. 386):

Definition 7 (Enterprise Architecture).

The enterprise architecture is referred to as an architecture defining and interrelating data, hardware, software, and communication resources as well as the supporting organization required to maintain the overall physical structure required by the architecture.

Only EA-based taxonomies are suited for the inspection of a neuronal technique use because they combine both modeling approaches and technical implementations (including data, hardware, software, communications, and methodology). Since architectures consider conceptual models of software systems within the business context, the following refers to the field of EA modeling and excludes definitions and philosophical discussions about EA. Furthermore, only-IT-related types of architectures, such as software architectures, system architectures, etc., are excluded as well because of their missing systematic link to the business context.

Considering more than 50 EA frameworks (Matthes, 2011, p. 57), the following focuses on a selection. According to Kotusev (2016), three kinds of EA frameworks can be identified, whose most important representatives are considered to have founded the CoNM:

1. Business Systems Planning (BSP) methodologies, which can be interpreted as the earliest origins of the EA. Representatives are the IBM's editions of BSP (IBM-Corporation, 1975; IBM-Corporation, 1978; IBM-Corporation, 1981 and IBM-Corporation, 1984). Further, there can be found IBM-BSP-similar approaches, such as Method/1 (Lederer and Gardiner, 1992), Martin (1982), and Nolan and Mulryan (1987). Company-specific, BSP-similar approaches are Adriaans and Hoogakker (1989), Davenport (1994), Lederer and Gardiner (1992), Lederer and Putnam (1986), Lederer and Putnam (1987), Sullivan (1985), and Zachman (1982). Here, IBM's BSP methodology of the first edition

- (IBM-Corporation, 1975) is selected from several representatives because it is the first approach that resembles EA in many important aspects (Kotusev, 2016, p. 30).
2. Early Enterprise Architectures (EEA), which can be interpreted as the very first EA approaches. Representatives are Wardle (1984), the *PRISM EA framework* (PRISM, 1986) and the editions of the Zachman framework (Zachman, 1987; Sowa and Zachman, 1992; Zachman, 1999). Further, there are the *NIST EA model* (Rigdon, 1989), the *architecture development methodology* of the Government Accountability Office (GAO, 1992), and the *Technical Architecture Framework for Information Management—TAFIM* (TAFIM, 1996b; TAFIM, 1996a). Here, the Zachman framework of the first edition (Zachman, 1987) is selected as he is considered the father of the EA and this model as the foundation of a modern EA (Kotusev, 2016).
 3. Modern EA, which can be interpreted as state-of-the-art EA approaches. Representatives are the *Federal Enterprise Architecture Framework—FEAF* (FEAF, 1999; FEA, 1999) and the *Command, Control, Computers, Communications, Intelligence, Surveillance and Reconnaissance—C4ISR* framework (C4ISR, 1997). Further, there are the *Department of Defense Architecture Framework—DoDAF* (DoDAF, 2007a; DoDAF, 2007b; DoDAF, 2007c; DoDAF, 2007d) and editions of the *TOGAF* standard (TheOpenGroup, 2011). Here, TOGAF is chosen from several representatives since it is the most frequently cited and the most frequently discussed approach (Simon, Fischbach, and Schoder, 2013). Moreover, it is the de facto industry standard in EA (Kotusev, 2016, p. 33).

Beside EA frameworks, the following approaches are considered:

4. Information system architectures, which provide a focus on business contexts, such as Architecture of Integrated Information Systems called ARIS (Scheer, 2002).
5. Computer Integrated Manufacturing Systems, which raise a claim to produce insights in regard to a common enterprise modeling, such as CIMOSA (AMICE, 1993; Vernadat, 1996; Vernadat, 1998).

The following paragraphs present the selected approaches alphabetically.

2.1.2.3.1 ARIS

Being an integrated concept for information system architectures, modeling methods, and modeling tools, Architecture of Integrated Information Systems (ARIS)

provides a framework with five levels (Scheer, 2002, p.112): On the *process design level*, a modeling and analysis software tool called ARIS-Platform (see section 2.6.1.2) is provided. It supports the creation of process models following the *ARIS House views* (Scheer, 2002, p.37). These models are used for process management activities, such as the creation of process models, product models, the specification of knowledge management, their assessment and benchmarking, the conductance of simulation runs, quality assurance, and the provision of models on behalf of process warehouses (Scheer, 2002, p.57). The *process planning and control level* provides tools for the monitoring and control of process models. The *workflow level* provides a product for prototyping workflows and their integration with further workflow systems so that models constructed can be implemented by staff and machines. On the *business object level*, generic standard objects for logistics are provided, which enable integration with standard software. The *technical environment level* characterizes technical components required for the embedding of former four levels, such as operating systems, data bases, and client-server interfaces, among others.

Considering all five framework levels, relevant perspectives of the *ARIS House* are concretized in five phases (Scheer, 2002, p.39): First, a *strategic reference solution* is created which connects required processes with company objectives and considers the provided infrastructures. Second, requirements about applications are defined focusing on the first framework level, which is the process design level. This phase is termed as *requirement definition*. In the third phase, models are modified so that they can be integrated with technical components, such as data base systems, network architectures, programming languages, etc. (*design specification phase*). Fourth, technical components are realized (*implementation description phase*). Fifth, as the entire information system is finalized, it will be used and maintained during (*run time*). The phase-wise concretion considers the five perspectives of the *ARIS House*, which refer to organization, data, control, function, and performance views (Scheer, 2002, p.37) and considers iterations for the rough *concept*, the more detailed *data concept*, and the *implementation*. All this leads to an *ARIS configuration*, which represents a possible solution for an information system supporting business process models in organizations.

Interim Conclusion: An integration with neuronal techniques cannot be identified at the ARIS framework architecture as follows:

- Neuronal modeling objects are not considered at the *process design level*, which includes their use in modeling tools and process meta-models.

- Neuronal modeling objects are further not considered in planning and controlling activities on ARIS's second framework level. This excludes their use in process simulations and optimizations using neuronal libraries. Hence, on the workflow level, neuronal networks are not integrated with non-neuronal workflows (further workflow systems).
- On the business object level, standard objects are not mapped to neuronal representations, and an integration with non-neuronal software (standard software) is not considered at all.
- Since the technical environment level focuses only on non-neuronal software, a technical environment is not optimized for the realization of neuronal networks.
- Phases identified to be useful for the realization of the *ARIS House* therefore do not consider the creation and use of neuronal networks.
- An *ARIS configuration* does not lead to a software-supported neuronal business process level integrated within the organization.

2.1.2.3.2 Business Systems Planning (BSP)

Being the earliest origin of EA, BSP methodologies provide a top-down planning approach using architecture notations for the creation of a formal description of the relationship between business and IT.

IBM's first edition of the BSP approach includes the following two dimensions (IBM-Corporation, 1975): The first dimension focuses on four perspectives and their relation: organization perspective, business process perspective, data perspective, and information system perspective. In a stepwise manner, seven phases are realized (second dimension): First, *business objectives* are identified. Second, *business processes* and *data structures* are defined. Third, the *existing IT infrastructure* is analyzed and fourth, the *desired information systems* are planned. Fifth, an *action plan* is created for the transformation of as-is systems to to-be systems. This is realized and results are subsequently *communicated* in the final phase.

Interim Conclusion: Overall, IBM's BSP approach provides seven general kinds of activities, which consider four kinds of perspectives. None consider an integration with neuronal techniques as follows:

- Models created do not consider neuronal perspectives, neither on the organization perspective nor on the business process perspective, the data perspective, or the information system perspective.

- None of the provided seven *phases* consider activities dealing with neuronal libraries, tools, or concepts.

2.1.2.3.3 Computer Integrated Manufacturing Open System Architecture—CIMOSA

As an integrated concept for computer integrated manufacturing systems (CIM) and a methodology following an open system architecture (OSA), CIMOSA provides a vendor-independent framework based on building blocks. These can be used separately or in combination because blocks are independent and provide standardized interfaces. Since CIMOSA was supported by the European Union Program called ESPRIT and was developed in various research projects; the following summarizes works by (AMICE, 1993; Vernadat, 1996; Vernadat, 1998).

The entire collection of blocks builds a three-dimensional lattice as follows. The first dimension called *stepwise generation* characterizes different perspectives on an information system and refers to the organization, resources, information, and functions. The second dimension called *stepwise derivation* characterizes three phases required for the model construction. Phases are similar to ARIS phases and are called *requirements definition modeling level*, *design specification modeling level* and *implementation description level*. The third dimension, called *stepwise instantiation*, characterizes three steps required for the individualization of concepts. A first step provides basic requirements and is called *generic instantiation*. A second step transfers generic building blocks in branch-specific requirements (*partial requirements*). A third step transfers branch-specific building blocks in company-specific requirements and is called *particular instantiation*.

Interim Conclusion: Overall, 48 kinds of building blocks are specified in the CIMOSA framework, but none consider an integration with neuronal techniques as follows:

- The generation dimension does not consider neuronal perspectives.
- None of provided three *derivation* manifestations consider neuronal libraries, neuronal tools, neuronal methods, or neuronal concepts.
- The *instantiation* dimension does not consider recommended knowledge carriers (Nonaka and Takeuchi, 1995), such as *individuals* and *groups*.

2.1.2.3.4 The Open Group Architecture Framework—TOGAF

Being a state-of-the-art EA framework, The Open Group Architecture Framework (TOGAF) provides an operational approach for the development of technical architectures. This includes the design, planning, implementation, and maintenance of information systems in modern enterprise architectures.

The TOGAF is based on two dimensions as follows (TheOpenGroup, 2011): The first dimension considers four perspectives on an EA called *domains*. The *business* domain can be interpreted as a result of business process modeling and focuses on strategic, organizational, and processual issues. This includes business capabilities such as knowledge-intensive activities. The *data architecture* domain specifies data and data relations required for the business processes to proceed in one common model. This includes the categories *information*, *information groups*, and *information needs*. The *application* domain focuses on the organization and administration of applications, including application interfaces required for the business processes to proceed. The *technology* domain focuses on IT infrastructure components required for the business processes to proceed.

The second dimension specifies a cyclic methodology called *Architecture Development Method* (ADM). It develops and optimizes EA models by one preliminary and eight cyclic *phases*. In an initial phase, preliminary issues are covered and the framework is introduced. The first cyclic phase called *architecture vision* specifies objectives and responsibilities. Thereafter, architectural models of the four domains are constructed. This describes the as-is and to-be situations of the enterprise. In a fifth phase called *opportunities and solutions*, a selection of transformation projects is identified that will improve the as-is situation project wise. The sixth phase (*migration planning*) characterizes the collaboration of transformation projects, which is controlled in a seventh phase (*implementation governance*). The eighth phase focuses on the collection of requirements and external influences for consecutive iterations. This phase is called *architecture change management* and closes the activity cycle with the *architecture vision* phase.

Interim Conclusion: Overall, the TOGAF provides nine kinds of methodological activities, which consider four kind of models. None consider an integration with neuronal techniques as follows:

- The *domain* dimension neither considers neuronal perspectives nor neuronal libraries, tools, or concepts.
- None of provided nine manifestations of the *ADM* refer to activities dealing with neuronal libraries, neuronal tools, neuronal methods, or neuronal concepts.

2.1.2.3.5 Zachman Framework

Although not a concept directly allowing the realization of process models as information system, the Zachman framework integrates modeling and methodology and provides a framework inspired by the information system architecture (ISA). Hence, it at least describes the implementation of process models by architectural components. The Zachman framework has been developed for more than thirty years and is seen as a foundation for the enterprise architecture management discipline (EAM). It is popular because of its simplicity (Kotusev, 2016). Its evolution is based primarily on (Zachman, 1987; Sowa and Zachman, 1992; and Zachman, 1999) and is an ongoing process.

The framework comprises two orthogonal dimensions as follows:

The first dimension represents repetitive modeling phases (here called *life-cycles*) and provides responsible roles. The *scope* phase is specified by the *planner*. *Enterprise models* are characterized then by *company owners*. A *system model* is designed by *designers* in a third phase. It is then implemented by *builders* who create *technology models*. Single *components* are realized by *sub-contractors* consecutively. The *functioning system* is finally used by *users*.

The second dimension represents modeling perspectives (here referred to as *description fields*) and provides interrogatives. The *data* perspective characterizes *what* kind of data is required. The *how* is characterized by the *function* perspective. *Where* a process is realized is characterized by the *network* perspective. The *person* perspective specifies *who* participates in a certain process. *Time* perspectives gather information about the *when*. *Why* is characterized by *rational* perspectives.

Each field of a matrix built by these two dimensions is characterized by a specific method that helps gather the required information systematically.

Interim Conclusion: Overall, the Zachman framework presents 36 kinds of modeling methods but none consider an integration with neuronal techniques as follows:

- The *perspective* dimension does not consider neuronal perspectives.
- None of the provided six *life-cycle* manifestations considers activities dealing with neuronal libraries, neuronal tools, neuronal methods, or neuronal concepts.

2.1.2.4 Critical Appraisal of Taxonomies

Faced with the aforementioned taxonomy approaches and its interim conclusions, it seems that today's knowledge and business process modeling approaches show numerous aspects. However, they are still immature and do not consider ANN techniques:

- None of the presented *perspectives* of perspective-oriented taxonomies consider neuronal perspectives or neuronal modeling objects and therefore, an integration with currently available perspectives is missing.
- None of the presented *approaches* of approach-oriented taxonomies consider neuronal perspectives or neuronal modeling objects, and therefore, an integration with currently available approaches is missing.
- None of the presented *architectures* of architecture-based taxonomies consider neuronal perspectives or neuronal modeling objects, and therefore, an integration with currently available architectural components is missing.
- None of the presented *architectures* of architecture-based taxonomies consider phases dealing with neuronal libraries, neuronal tools, neuronal methods, or neuronal concepts. An integration with currently available concepts, tools, or libraries is therefore missing as well.

A research gap becomes evident here.

A synthesis of the approaches presented here, enriched with ANN techniques, will support a broad acceptance of the CoNM in the corresponding research communities. Further, a synthesis ensures its extensive application possibilities. The idea for a ‘holistic’ synthesis efficiently representing all modeling perspectives and being applicable in all modeling situations is not new (Kueng et al., 1996, p. 98; Giaglis, 2001, p. 14; Sultanow et al., 2012, p. 3310). Experiences show that the multiplicity of possible modeling goals and objectives lead to impractical modeling approaches. Probably, a synthesis generates complex models reducing the ease of use for any realistic setting (Curtis, Kellner, and Over, 1992).

Hence, for the construction of a CoNM, a compromise between holism and practicability must be identified. In principle, a practical and holistic synthesis must address the following aspects:

- Similar to Beer (1979), it must broadly consider business processes and consider their operational behavior in regard to managerial co-ordination, controlling, development, and policy.
- According to Kueng et al. (1996, p. 98), it must describe the process of modeling. Here, it must become clear what kind of information is required as input for the methodology and is produced as output from its application.
- Inspired by Fröming (2009, p. 3310) and (2012, p. 3310), a knowledge and knowledge-flow-oriented dimension must be considered.
- In accordance with Scheer (2002, p. 112), it must not be limited to the design of processes but should also consider tools, methods, and further architectural components.

- ANN techniques must be harmonized with the understanding of the business process modeling and knowledge modeling.

The creation of a CoNM will therefore consider the aforementioned points, and it will be based on a practical synthesis of the perspectives presented here.

2.1.3 Process Modeling Languages

Considering the basic process of modeling concepts and definitions (section 2.1.1) as well as criteria and perspective-based standards from process taxonomies (section 2.1.2), the following examines prominent notations for the representation of process models, what is termed as *process modeling languages*.

Based on the construction-oriented and system theoretic definitions of process models and definitions about the process model creation (Def. 5), process modeling languages are defined as follows:

Definition 8 (Process Modeling Language).

A process modeling language (synonym for process modeling notation) is, not necessarily but mostly, a graphical notation satisfying more or less sophisticated syntactic rules to be applied by a model creator to support the construction of a model creator's reality and to create a representation of its subjective or objective reality about processes. What is known as the process model is within the creator-specific subjective or objective model environment on arbitrary levels of abstractions and granularities of deep structures.

As the definition does not exclude specific process models, it includes the specialization of domain-specific process models, such as business process model or process models for (artificial) neuronal networks. Here, each domain demands individual process modeling elements in coherence with the most basic and abstract meta-model of deep structures available (see Fig. 2.2). Although this relation is not visible when considering a specific process model example, and most process modeling notations do not satisfy this formalization adequately because they only address very specialized issues and only a small selection of common modeling perspectives (cf. section 2.1.2), this relation is essential as it holds the modeling world together.

According to the *modeling definition* (Def. 5), during the act of process modeling, knowledge is considered by the model creator as follows: First, knowledge is considered to abstract from a sophisticated reality by the model creator (see *forget-*

ful mapping of Fig. 2.1). Second, knowledge is considered to complement models with attributes (see *extension mapping* of Fig. 2.1). Third, knowledge is considered by the model creator to efficiently and appropriately express representations (see *bidirectional attribute mapping* of Fig. 2.1) on behalf of the corresponding process modeling language and its conventions. Since these kinds of knowledge are available within the model constructed in a tacit manner, the subjectivity and the hardly comprehensible view of the model creator leads to a complicated tension between original creator-model relationship as presented in Fig. 2.5.

The term *mostly* in the definition implies that processes can be carried out by a combination of *graphical modeling items*, but can also refer to *mathematical expressions* or a corresponding representation as *script language* in the sense of computer-aided software engineering (CASE) as well (Case, 1985; Bergin, 1993; Bucci et al., 1994; Lang and Duggan, 2001). The common denominator here is the use of language constructs, such as *words* and *grammar*. By following the transformation rules, the models following different notations or representation forms can be *translated* into one or the other language. As the corresponding two models are well formalized, this translation can be carried out without information loss or information writing, which refers to the same level of *information entropy*.

Selection of representatives: As many process modeling notations can be found in the literature, the selection of adequate works must be addressed. Attempts searching for best candidates refer to the following e.g. The Business Application Research Center (BARC) Würzburg focuses on an analysis of EPK, BPMN, Petri-Nets, UML, and Entity-Relationship-Models as process modeling notations, Business Process Execution Language for Web Services (BPEL4WS), and XPDL as process execution languages (Böhn, 2007; Böhn, Burkhardt, and Gantner, 2010; Böhn et al., 2014). Further, the Fraunhofer Institute presents a synopsis of the most famous modeling languages with regard to company pragmatics and research literature, which refers to the BPMN, BPEL, EPK, IUM, Kommunikationsstrukturanalyse, Line of Visibility Enterprise Modeling LOVEM, Semantic Object Model SOM, Unified Modeling Language UML, PACE, and Ishikawa (Drawehn et al., 2014, p. 38). The contribution therefore selects process modeling languages according to the following criteria.

1. Modeling languages having a broad market distribution, which was determined by market research studies (Drawehn et al., 2014)
2. Modeling languages having broad acceptance in the modeling community
3. Modeling languages applied in at least one scientific publication

Adequate examination level: In order to satisfy requirements for an adequate management of objects of investigation (OoI) in the terms of SLR (Levy and Ellis, 2006), which corresponds to the appliance of Bloom's taxonomy (Bloom et al., 1956), each modeling language is described by the following categories:

- The short *abstraction* serves as a kind of introduction to modeling languages on the same, abstract level.
- Its individual *description* characterizes the individual OoI and is essential for the evaluation.
- *Side facts* issue the OoI-specific development and historic evolution. These are essential for the contextual embedding of the OoI, but not for evaluation.
- The provision of a *meta-model* visualizes modeling items and its underlying syntax. This is completed by *Modeling conventions* clarifying how the meta-model of the OoI is applied so that concrete process models are constructed.
- An *example* visualizes the appliance and underlines modeling strengths of the OoI.
- Finally, the *critical appraisal* evaluates whether and how the individual modeling language is suitable as a foundation for the CoNM.

Therefore, according to Bloom's taxonomy, categories considered for any modeling language correspond to the levels of *knowing* and *comprehending* if OoIs are described and side facts are presented, *applying* as a meta-model is generated and an example is presented, *synthesizing* as a common level of understanding is worked out, as well as *analyzing* and *evaluating* as common analysis criteria are issued and the process modeling languages are regarded with respect to the CoNM.

2.1.3.1 Petri Nets

The **Petri net** is a bipartite graphical modeling language, which enables the construction of mathematical-based models of cooperating processes. Therefore, it particularly focuses on discrete event-based distributed systems but enables the construction of process models showing the ordered temporal and logical sequence of activities. Therefore, it is suitable for the modeling the behavioral perspective of a system (section 2.1.2.1).

Description: Over the long period through which Petri nets have existed, an extensive range of Petri net variations have been designed. Only the bibliography of the University of Hamburg itself provides more than 8.500 entries (University of Ham-

burg, 2019). The following presents the most important five types following Balzert (2009).

The very basic type is called *Condition/Event Net* (C/E Net), which refers to directed graphs consisting of only two types of nodes: *places* representing states and being visualized by circles as well as *transitions* representing the transition from state to state visualized by rectangles. Edges linking the iterating modeling items of places and transitions refer to directed arcs and specify the *flow relation*. Thus, a Petri net can be defined as follows (Schuster, 2012):

Definition 9 (Petri Net).

A Petri net PN is a triple $PN = (S, T, F)$, for which holds:

- i S is a finite set of places,
 - ii T is a finite set of transitions,
 - iii $T \cap S = \emptyset$ and $T \cup S \neq \emptyset$,
 - iv F is the flow relation and describes the finite set of edges
- $$F \subseteq (S \times T) \cup (T \times S).$$

If initial markers are considered in the Petri net model additionally, these are called *tokens*, a system can be specified. These items enable the modeling of the dynamic behavior of the system during a simulation run at any point in a period of time. For the very basic C/E Nets, the following applies:

Definition 10 (Petri Net System).

A Petri net system PNS is a tuple $PNS = (S, T, F, M_0)$, for which holds:

- i (S, T, F) is a Petri net,
- ii the initial token is a function $M_0 : S \rightarrow \mathbb{N}_0$,
which assigns each place a number of tokens.

For the application of transitions during a simulation run, the environment of any element is characterized by the following:

Definition 11 (Petri Net Element Characterization).

For any element $x \in S \cup T$ of a Petri net $PN = (S, T, F)$ holds:

- i $\cdot x = \{y : y \in S \cup T \wedge (y, x) \in F\}$ the pre-range of x ,
- ii $x \cdot = \{y : y \in S \cup T \wedge (x, y) \in F\}$ the post-range of x ,
- iii $U(x) = \{\cdot x \cup x \cdot\}$ the environment of x .

The dynamic behavior enfolds as a condition for a transition to be fulfilled and realized. This is formalized by *transition rules*, which specify how tokens are routed by the *PNS*. Thus, for simple C/E nets, the following rules can be defined:

Definition 12 (Transition Rules In C/E Nets).

A transition t of a Petri net $PN = (S, T, F)$ is activated if

- i all places in $\cdot t$ provide tokens and
- ii all places $t \cdot$ do not provide tokens.

The activation of a transition removes tokens from any place of the pre-range and equips any place with tokens of the post-range.

More sophisticated Petri nets consider the following: First, *capacities* per place (Best, 1987; Desel and Reisig, 1996; Desel and Reisig, 1998), which extend the boolean state of places of the basic type of Petri nets and are thus called Place/Transition Nets (P/T Nets). Second, different kinds of transfer conditions and transfer objects (Jensen, 1996). Since these can be visualized by different colors or the use of numbers, these kinds are called Colored Petri Nets (CPN). Third, Petri nets considering an understanding of time (Popova-Zeugmann, 1991; Sloan and Buy, 1996; Jiroveanu, De Schutter, and Boel, 2007). This considers *atomic* events or transitions, which are realized without time consumption, and *complex* events or transitions, which demand the persistence of tokens for a certain period of time ($t > 0$). Here, a *deterministic* or *stochastic* time consumption can be applied. Fourth, Petri nets being modeled on behalf of a hierarchy of models (Fernández et al., 2008; Hong et al., 2012; Padberg, 2018), which does not exclude previously mentioned types of networks or their combination, such as hierarchical timed-colored Petri nets (Jensen, 1996). Here, special rules focusing on the consistency of the model must be considered. Fig. 2.6 presents the stepwise extension for the most important types of Petri nets; more sophisticated Petri nets are built on simpler variants in providing reinterpretations of basic modeling items (Balzert, 2009). Particularly, the representation of Petri nets on the basis of *matrices* are suitable and can be used for mathematical operations as they are required for the activation easily (Marsan et al., 1994).

Since the complexity increases drastically because of bigger network topologies and more sophisticated Petri net approaches, the attempt to answer analytical questions becomes challenging. Examples can be found in the following works: Jones, Landweber, and Lien (1977), Ruiz, Frutos-Escríg, and Cuartero (1991), Chiola et al. (1993), Esparza (1994), Esparza (1998), Esparza and Heljanko (2008), Wimmel (2008), Priese and Wimmel (2008). Only the simple C/E Nets themselves lead to 2^n possible states having n places, which demand very laborious and con-

Modeling Item		Type of Net				
Designation	Graphical Representation	Condition/Event Nets	Place/Transition Nets	Colored Nets	Time-based Nets	Hierarchical Nets
Place		Condition <i>marked or not marked</i>	Place <i>Capacity > 0</i>	Predicate <i>Capacity > 0</i>	Place <i>Time Interval</i>	Channel
Transition		Event	Transition	Event <i>Switching Condition & Switching Effect</i>	Transition <i>Time Interval</i>	Instance
Arcs		Unweighted	Weighted	Weighted <i>Constant and Variable Annotation</i>	Depending on Type of Net	Depending on Type of Net
Tokens		Token <i>Uniform</i>	Token <i>Uniform</i>	Object <i>Individual</i>	Depending on Type of Net	Depending on Type of Net

Figure 2.6 Overview of important types of Petri nets (synthesized from literature)

fusing analyses. As an alternative to these analyses, a simulation can be used to find answers on behalf of networks constructed (see section 2.3).

Side facts: The research about Petri nets was initiated by Carl Adam Petri from the *Faculty of Mathematics and Physics* of the University of Darmstadt in 1962 as a semi-formal modeling concept (Petri, 1962). Since then, various Petri net variants have been designed. Approaches such as EPC (see section 2.1.3.2) and UML (see section 2.1.3.3) have been built on Petri net-based concepts and tools particularly focusing on the use of Petri nets have been developed (University of Hamburg, 2019).

Meta-model: Fig. 2.7 provides the general *Petri net process meta-model*. Here, it becomes clear how extensions of different Petri nets are integrated by annotations, weights, and variables. Since the Petri net nomenclature is formerly well specified, models can be constructed based on a very little set of modeling items.

Considering the presented general business process model of Petri nets, the process models are constructed by the following design rules (Dumas, Aalst, and Hofstede, 2005):

1. Petri net's core nodes refer to *places*, *transitions*, *edges*, and *tokens*.
2. The transition's denomination should reflect its unique characteristic as a point in time and is represented by a rectangle.

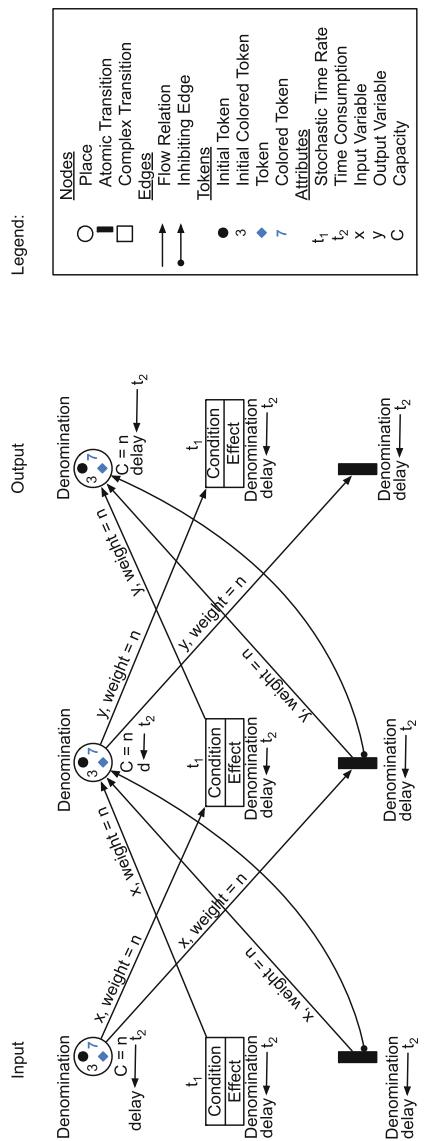


Figure 2.7 Generic Petri net process meta-model (synthesized from literature)

3. The place's denomination should reflect its structural perspective as an object or state required to accomplish the task.
4. Edges connect two elements corresponding to the sequence of action so that they are directed.
5. Places and transitions might not be the predecessors or successors of the same element type. So, Petri net-based process models comprise alternating elements of *places* and *transitions*.
6. Tokens represent the system's initial and current state.
7. For the transition, the required information is positioned adjacent to the corresponding modeling item. This includes, for instance, capacities and delay time information for places as well as weights and variables for edges.

Example: An example model for a Petri net is visualized in Fig. 2.8. It shows the production of cyber-physical workpieces facing two alternative conveyor routes and thus highlights the Petri net conventions.

Since the model's purpose was to visualize the decision identification for the rooting of workpieces, production steps before and after the bottle neck presented are outside the modeling scope. The process initiates with the transportation of work-

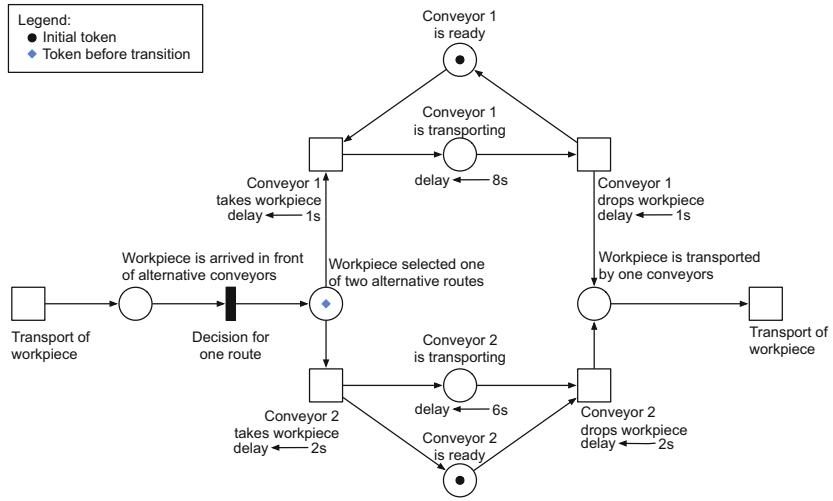


Figure 2.8 EPC-based process model example (created to show the strength of the object of investigation)

pieces which must instantly decide which route to take. As two alternative conveyors are part of the production setting, two triangle loops present their resource allocation. Only if the *conveyor is ready* and a *workpiece has selected* the corresponding conveyor route will the *conveyor take the workpiece* and be blocked for transportation tasks during the delay specified. After the period for *taking*, *transportation*, and *dropping* has passed, the workpieces are dropped so that the *conveyors are ready* for further transportation tasks. Since the sum of delayed times of both conveyor routes is ten seconds, the choice of the workpiece is trivial, and a simulation presents the question regarding whether two conveyor routs are sufficient to prevent blockage of the entire production process at all. As the model does not specify which route is chosen by the workpiece, the example presents a non-deterministic decision process.

Critical appraisal: The Petri nets provide only very simple items for the construction of process models. While this limits the flexibility to construct process models of specialized domains, it clearly supports the interpretability of the models constructed because of the strict application of formalized concepts. Therefore, before the use of Petri nets in concrete domains, complicated extensions have to be realized. For instance, a focus on business processes or organizational context have not been identified so far. Further, criteria focusing on the modeling of knowledge, information, business, or (artificial) neuronal processes are not satisfied and demand for laborious extensions. Thus, only criteria issuing the behavior perspective are satisfied.

None of modeling items refer to biological compartments of NN or ANN, but the Petri net concepts provided at least consider the mathematical concepts Petri nets are suitable for integration with neuronal mechanisms. Because of their strict formalization, they are very well suited to the specification of process simulation settings and the visualization of the simulation runs. Depending on the complexity of the simulations intended, for e.g., the number of conditions, the number of different modeling objects, and the level of hierarchies, circumstantial models and thorough implementations are required. Hence, modeling language criteria from the simulation domain are satisfied partly, and the modeling language is effectively prepared for tool-based implementations.

Unfortunately, the use of real-world positions and augmented reality concepts, as required for a CoNM, is not supported by Petri nets. Owing to their formalization, they are very well prepared for modeling tools and augmentations can indeed be extended. Further, modeling tools can simply provide rule checking approaches, thereby enabling the comfortable and formal correct model construction. A simulation tool can provide simulation engines building on modeling concepts of Petri nets easily and ANN tools can integrate Petri net's mathematical concepts with concepts

required by ANN. Therefore, if Petri net extensions are provided for corresponding modeling domains, a combination with ANN tools is suitable as the basis for the construction of a CoNM.

2.1.3.2 (e)EPC

The Event-driven process chain (EPC) is a graphical modeling language, which enables the construction of process models showing the ordered temporal and logical sequence of activities. Thus, it is suitable for modeling the behavioral perspective on a system (section 2.1.2.1). Since EPC-based process models serve as integrative components of the *House of ARIS* (Scheer, 2002, p.37) and link its four aspects, these are essential to EA model specification of the ARIS Framework (section 2.1.2.3.1).

Description: According to Scheer (2002, p.41), the EPC integration role is visualized in Fig. 2.9.

In the figure, five views are visible, integrated by the *control view*. The created EPC-based process models are complemented with organizational entities (e.g. staff, departments, sites) defined on behalf of models of the *organizational view*, data entities (e.g. text documents, calculation files, and offers) defined on behalf of models of the *data view*, and performance entities (e.g. of products, services) defined on behalf of models of the *performance view*. Process activities connecting these entities are

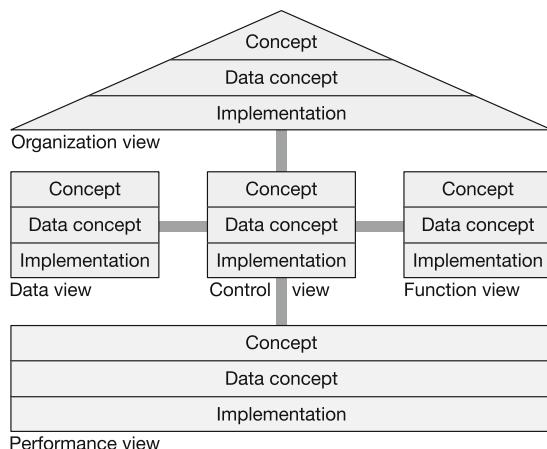


Figure 2.9 House of ARIS (Scheer, 2002)

selected from models defined in the *function view*. Hence, they are called *functions* from the EPC perspective.

Each view provides a collection of entities used in process models to be created. Typically, the views surrounding the control view are provided as hierarchical arrangement, so that entities can be provided on different levels of abstraction. As models of all views are defined iteratively, the models of a rough *conceptual* iteration, more detailed models of a *data concept* iteration, as well as detailed models of an *implementation* iteration can be related to the models of other views (Scheer, 2002, p.41).

Side facts: EPC was initially developed by Prof. August-Wilhelm Scheer from the *Institute for Business Informatics* of the University of Saarland in 1992 as a semi-formal modeling concept (Scheer, 2002). Since then, various approaches have provided attempts for formalizations and process instantiation (Aalst, 1999; Nüttgens and Rump, 2002). Further, various EPC variants and extensions have been developed, which are called extended Event-driven process chain (eEPC). These provide further modeling entities, business objects, organizational units, and modeling views without changing the original EPC concept (Aalst, 1999; Scheer et al., 2003; Mendling, Neumann, and Nüttgens, 2005; Davis, 2012). Example items can be found in modeling items for databases, customer data, risks, objectives, empty connectors, multiple instantiations, cancellations, and example views refer to risk diagrams, IT system diagrams, access diagrams, and collaboration diagrams.

Meta-model: Fig. 2.10 provides the general *ARIS business process meta-model*. Here, it is apparent how modeling objects of different views are integrated by function elements. Since the basic EPC nomenclature only visualizes the core elements, models seem clean and easy to interpret. More sophisticated eEPC-based models therefore seem more complex.

Considering the aforementioned general business process model of ARIS, EPC-based process models are constructed by the following EPC design rules (Keller, Nüttgens, and Scheer, 1992; Aalst, 1999; Dumas, Aalst, and Hofstede, 2005):

1. EPC's core nodes refer to *activities*, which are sometimes called *functions*, *events*, and logical *connectors*.
2. The event's name should reflect its unique characteristic as a point in time and is represented by a hexagon.
3. The activity's name should reflect the time-consuming aspect of the accomplished task.

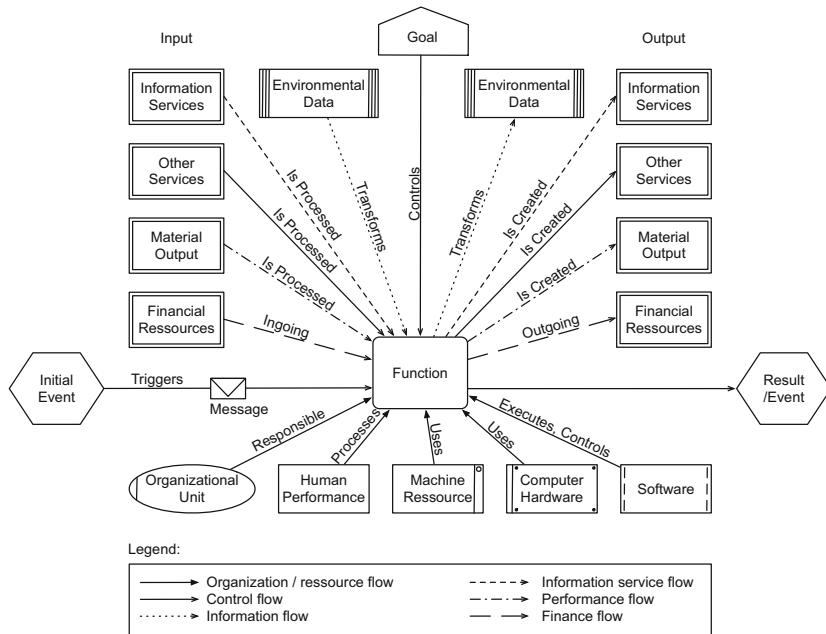


Figure 2.10 Generic ARIS business process meta-model (synthesized from literature)

4. Connectors are represented by a circle and its corresponding symbol and can be divided into into upper and lower parts so that one can differentiate between incoming and outgoing rules. Here, one can find the *AND rules*, the *OR rules*, and the *XOR rules*.
5. EPC-based process models start and end with an *event* or *process interface* so that one can define when a business process is intended to begin and end.
6. EPC-based process models at least comprise one atomic activity or a composition of further EPCs.
7. Edges connect two elements corresponding to the sequence of action, so they are directed.
8. Each event and each activity has only one incoming and one outgoing edge.
9. Events and activities might not be the predecessors or successors of the same element type. Thus, EPC-based process models consist of the alternating ele-

ments of *event* and *activities*. In between, they can provide connectors as part of the control flow.

10. eEPC-based process models relate documents as input and output objects with a *data flow*.
11. eEPC-based process models relate organizational units as input and output object with a *associations*.

Example: An example model for an eEPC is visualized in Fig. 2.11. It shows the IT service process inspired by the ITIL standard and thus clarifies EPC conventions.

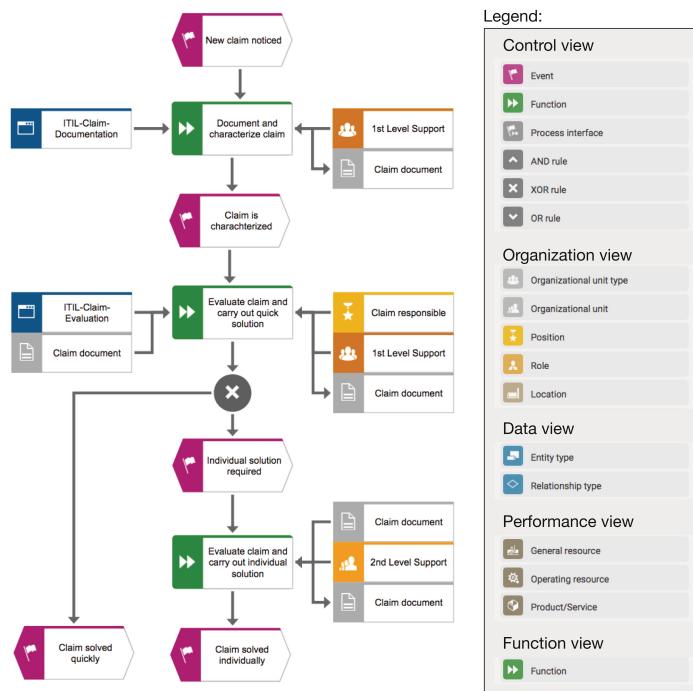


Figure 2.11 EPC-based process model example (created to show the strength of the object of investigation)

Here, one can notice the process model on the left. The legend on the right provides the current shape style for modeling items provided by state-of-the-art

software tools such as *ARIS Cloud*. Here, they have been clustered by the five corresponding ARIS views. Considering the various approaches for eEPC, the mentioned items refer to a selection of only some and the example visualizes their exemplary interplay and the application of EPC design rules.

Critical appraisal: The EPC provides standard items for the construction of business process models. In particular, its extensions provide nearly any item required for a process modeling and even the visualization of live process instances along with their monitoring and realization on behalf of workflow management systems. Therefore, basic modeling language criteria from the business process domain can be satisfied.

Regrettably, modeling items for different kinds of knowledge and different forms of conversions are not provided. At least, *documented knowledge* and *knowledge categories* can be placed within available process models. Since a knowledge view is not provided, an adequate knowledge management cannot be performed. So, modeling language criteria from the knowledge management domain are not satisfied.

Since a simulation view is not provided, only basic simulation items can be considered in EPC-based models. On behalf of probability assignments or function-specific parameters such as costs, durations, or occurrences, and so on, simple analyses or simulation runs can be realized. These are far away from position-based or complex simulation runs and their animation. However, a differentiation of process instances and process schematics is not enabled, which is essential for simulation runs. Therefore, the specification of simulation runs cannot be realized on behalf of graphical modeling objects. Additionally, it is not possible to model a hierarchy of simulation scenarios including the acquisition of simulation data. Hence, modeling language criteria from the simulation domain are not satisfied on an overarching base.

As not any modeling item refers to biological compartments of NN or ANN, criteria from the ANN domain are not satisfied by the modeling language EPC at all.

As approaches formalizing the EPC are available, their use in software tools is optimal. The simple rule checking carried out by tools enable comfortable model construction, such as the tool called ARIS-Platform exemplifies (see section 2.6.1.2).

2.1.3.3 UML 2's Activity Diagram

The Unified Modeling Language (UML) is a collection of graphical modeling languages, which visualize the design of a system and includes the modeling of processes (Balzert, 2009). Hence, it is suitable for the modeling of the behavioral

perspective of any system and considers additional objects of the organization perspective (section 2.1.2.1).

Description: In total, the UML 2.0 defines 14 types of diagrams. Here, one can find the following three diagram clusters. First, structure diagrams that visualize the static structure of systems, second, behavior diagrams that visualize the behavior of a system and third, diagrams that visualize different aspects of interactions. A characterization of each type can be found in Fig. 2.12.

Since UML models follow the XML Metadata Interchange (XMI) format, they can be exchanged among UML tools easily and UML diagrams are well prepared for the modeling, simulation, and process execution (Object Management Group, 2015). As only *activity diagrams* provide a focus on processes and workflows and other kinds of diagrams focus on models required particularly for software design, the following focuses on the analysis of *activity diagrams* only.

Side facts: The UML was initially developed by Grady Booch, Ivar Jacobson, and James Rumbaugh at *Rational Software* in 1994–1995 (Booch, Rumbaugh, and Jacobson, 2005) by combining individual approaches such as the Object-modeling technique (OMT, by Rumbaugh et al., 1991), the Object-oriented design (OOD, by Booch, 1990; Booch, 1994), and the Object-oriented software engineering (OOSE, by Jacobson, 1992). In 1997, the development was overtaken by the *Object Management Group* (OMG). Since then, various UML revisions have been released and in 2005, it was confirmed as approved standard by the *International Organization for Standardization* (ISO), which refers to the *UML* of vers. 1.4.2 (ISO Central Secretary, 2005). Since its most current version refers to the *UML* of vers. 2.5.1 (Object Management Group, 2017), the following considers the most modern version.

Meta-model: Fig. 2.13 provides the general *UML business process meta-model* for activity diagrams. Here, it is evident how the behavioral perspective of a system is modeled on the basis of Petri net-like semantics.

Notational elements such as *activity edges* have been excluded in this visualization since they only refer to issues not affecting the underlying model but notational issues (Object Management Group, 2017, p. 380).

Considering the presented general business process model of the UML, UML-based process models are constructed using the following UML design rules (Object Management Group, 2017, p. 379):

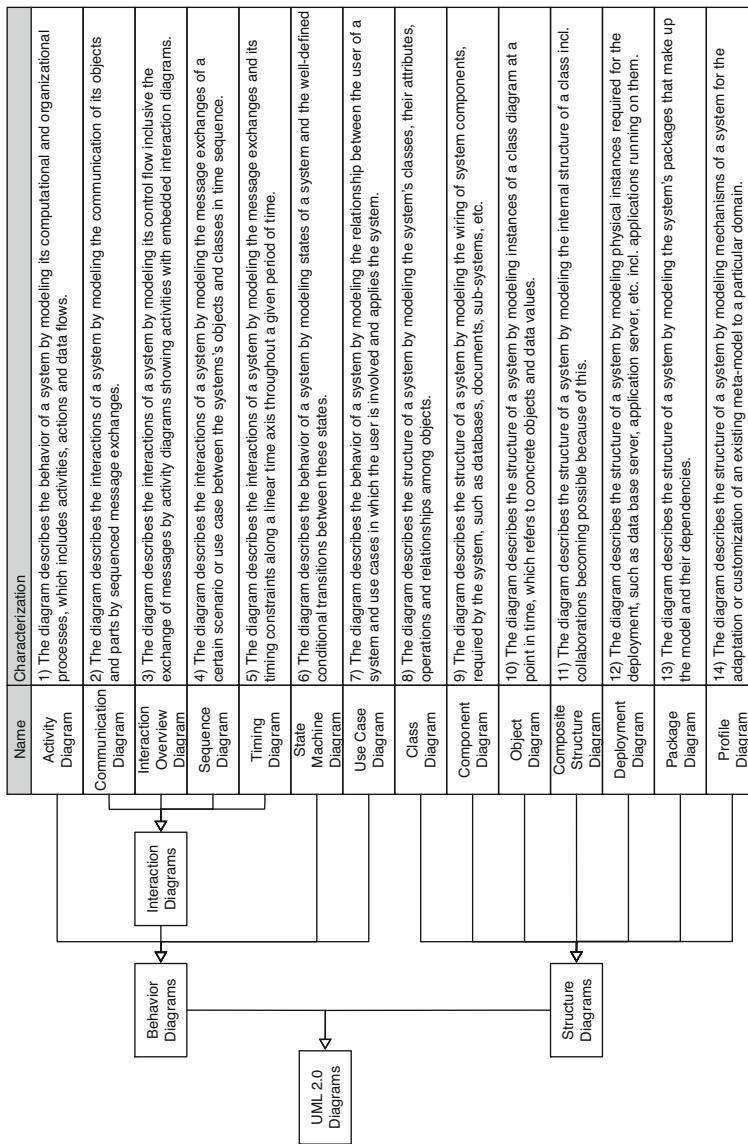


Figure 2.12 UML diagram types (synthesized from literature)

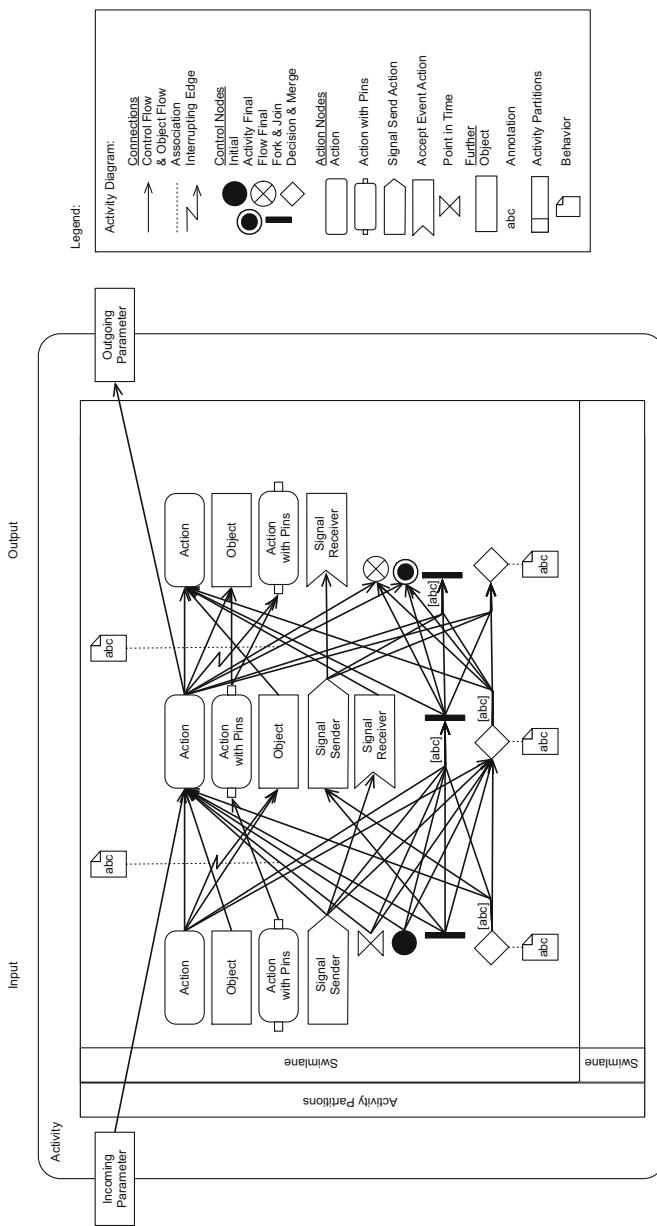


Figure 2.13 Generic UML business process meta-model (synthesized from literature)

1. UML activity diagram's core node refers to *actions*, *objects*, and *control nodes*.
2. The action's name reflects its time-consuming perspective of the accomplished activity.
3. The diagram is initiated with an *initial control node* and finished by an *activity final*.
4. *Flow final* control flow indicates the end of an *object flow* during the activity.
5. The diagram depicts four kinds of actions. First, simple *actions* refer to the smallest execution unit within the diagram. Second, *action with pins* indicate the reception and/or sending of objects. Here, incoming objects are visualized by a small rectangle on the left of the action and outgoing objects are visualized by a small rectangle on the right of the action. Third, *signal sending actions* refer to actions transmitting a signal at completion. Fourth, *accept event actions* refer to actions beginning when a signal is received.
6. *Actions* are linked by *activity edges* modeling the control flow.
7. *Activity edges* may be named so that individual annotations or weights can be associated.
8. The control flow can include *control nodes*, such as *forks and joins* for parallel control flows and *decision and merges* for conditional control flows.
9. *Events* can be modeled by *accept event actions*, which wait for the reception of the event, or by the occurrence of a certain *point in time*.
10. Either two object flow arrows link an *object* with a subsequent and a preceding *action* or one object flow arrow links the object node pins of two actions.
11. The realization of an action can be specified by a *behavior* item associated with the object flow. This includes specifications about a *selection* or *transformation*.
12. A decision's identification can be specified by a *behavior* item associated with the decision or merge node.
13. Actions having a common denominator can be grouped by *activity partitions*, which can comprise a hierarchy of swimlanes. Therefore, a swimlane can represent organizational units, roles, persons, and so on.
14. An *interrupting edge* connects two *actions* in order to indicate that a prior action can be interrupted and to specify the manner of proceeding if the action is interrupted.

Example: An example model for an UML is visualized in Fig. 2.14. It shows the process called *proceed order* with the help of the activity diagram and so clarifies the application of UML conventions.

Here, it is evident that the activity receives an input object called an *order* sent by a customer, for instance, and produces an output object called a *package*. In between these two objects, actions can be found that are realized on behalf of the three roles

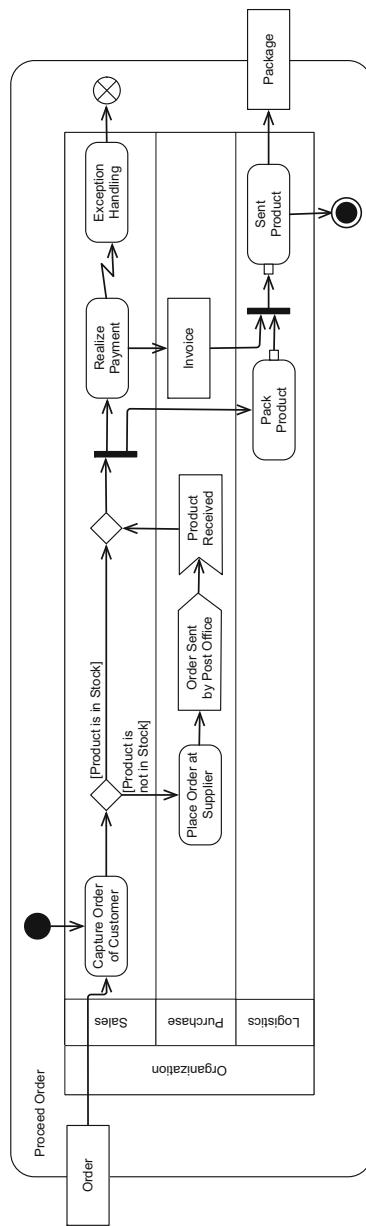


Figure 2.14 UML-based process model example (created to show the strength of the object of investigation)

sales, purchase, and logistics. While the sales department captures the order and realizes the payment, a purchase is realized by the purchase department if a product is not in stock. Here, the process is continued as a signal is received, which refers to the arrival of the product ordered. If the payment fails, an *exception handling* action is conducted that leads to an individual flow final. The logistic department prepares the packages and sends the product.

Critical appraisal: The UML activity diagram has its strengths in terms of the modeling of processes required for software systems. Therefore, common process modeling mechanisms required from the behavior and function perspective can be satisfied, but a focus on business process context is not provided at all. Here, available items such as the *swimlane* can be associated by the interpretation of the model constructor to visualize organizational entities e.g. Unfortunately, this results in conflicts in overlapping interpretations and corresponding modeling language criteria not being satisfied. Therefore, the UML modeling concept is only suited for modeling tools focusing on simple processes and more specialized contexts demand for very laborious implementations.

Because of the focus on software domains, the UML is suited to specify simulation models with high efforts. Although elements are rare, probabilities or further attributes are allowed to be associated with edges or nodes. Since this is not strictly formalized, only simple simulation runs can be realized by complicated human interpretation. This does not satisfy the requirement to map process models to computational models automatically. Thus, the UML modeling concept is only prepared for tools realizing simple simulation runs. Simulation tool criteria, even for complex simulation settings, are therefore not satisfied.

As none of the modeling items required by an adequate knowledge management is provided by the UML directly, objects can be interpreted to represent information, knowledge, and so on. Since this interpretation is laborious and involves many conflicts, the basic concept of UML does not enable the construction of process-related knowledge models. Thus, the corresponding modeling language criteria from the information and knowledge perspectives are not satisfied at all. They demand thorough implementations here.

Since neither modeling items nor ANN techniques are considered by the UML modeling concept, no diagram satisfies modeling items from the ANN domain or criteria required by ANN tools.

Since the entire UML concept provides neither a focus on knowledge transfers nor on ANN, the concept only applies to the context of simple processes. Therefore, it is not suitable for a combination with ANN tools as a basis for the construction of a CoNM.

2.1.3.4 BPMN

The Business Process Modeling Notation (BPMN) is a graphical modeling language, which enables the documentation of business processes and workflows (Rosing et al., 2015). As the BPMN provides a sizeable collection of semi-formal and weakly defined modeling symbols, it is suited to human experts instead of computer-based simulations or algorithm-based approaches to construct process models. Thus, it is suited to the human-based modeling of an interpretable behavioral perspective of a system and considers objects of the information flow-oriented perspective (section 2.1.2.1).

Description: The BPMN of the most current version provides three kinds of models (Object Management Group, 2013): First, *Business Process Diagrams* (BPD), which offer a single perspective on business processes, second, *Choreography Diagrams*, which group interactive process activities of two process participants, and third, *Conversation Diagrams*, which group process communication activities of process participants. The three diagrams are specified as follows.

(1) In the BPD, processual components refer to activities represented by rounded-corner rectangles of the following four types (Rosing et al., 2015). First, *tasks* represent the atomic unit of work and jointly represent higher level procedures. Second, *sub-processes* group a collection of tasks in order to represent them on a more abstract level of detail. This link is visualized by a plus sign. Third, *transactions* group activities, which must be treated as a whole. If its activities are not realized entirely, they must be undone. Fourth, *call activities* indicate that parts of a global process are reused. While the third is visualized by a doubled line, the fourth is indicated by a thick line. Generally, these kinds of processual components are known as *activities*. (2) The choreography diagram presents processual components as rounded rectangles having its process participants on top and bottom, focusing on their interaction on the basis of message exchanges. Thus, they are called *interactions*. As interactions can consider atomar interactions and sub-interactions, the diagram can build a hierarchy of interactions. (3) In the communication diagram, processual components refer to message exchanges by process participants of two types: While the first defines a set of logically related message flows, the second refers to a globally defined and re-usable conversation.

The BPD and choreography perspective further provide events represented by circles and referring to three types: First, *start events* indicate the start of a process. Second, *intermediate events* can occur during the realization of a process. Third, *end events* indicate the end of a process.

Additionally, the behavior of the process system modeled is specified on behalf of a collection of *gateways*. A rich collection of pictographs broadens this set of

modeling shapes significantly since basic items might be combined with individual pictographs. E.g. a *clock* symbol or a *message* symbol is allowed to be set inside basic event shapes or inside basic gateways so that an event is happening or gate is activated if a certain time of day or number of messages is reached.

Although the BPMN diagrams are not prepared for the modeling, simulation, and process execution, these activities are simplified since the BPMN diagrams are specified by XML-based diagram structures. Thus, the exchange between various modeling tools is simplified (Simpson, 2004).

Side facts: The BPMN was initially developed by Stephen A. White of the IBM and published by the *Business Process Management Initiative* (BPMI) in 2004 (White, 2004). Subsequently, the development and maintenance was taken over by the *Object Management Group* (OMG) in 2005 (Chinosi and Trombetta, 2012) and various versions of the modeling language have been released (Object Management Group, 2019) since: the BPMN of vers. 1.1 in 2008, vers. 1.2 in 2009, and vers. 2.0 in 2011. Since the most current version refers to the *BPMN* of vers. 2.0.2 (Rosing et al., 2015), the following considers the most modern version.

Meta-model: Fig. 2.15 provides the general *BPMN business process meta-model*. It is apparent how incoming (on the left) and outgoing (on the right) modeling objects are related in the BPMN.

Considering the presented general business process model of the BPMN, BPDs are constructed by the following BPMN design rules (Object Management Group, 2013):

1. BPD's core node refers to *activities*, *events*, and *control nodes*.
2. The activity's name should reflect the time-consuming aspect of the accomplished activity.
3. BPMN-based process models are initiated with a *start event* and end with an *end event*.
4. Within BPD, *interim events* can affect the process realization.
5. Each BPD at least consists of one type of activity, which refers to a *task*, a *sub-process*, a *transaction*, or a *call activity*.
6. Control nodes are represented by rectangles as well as their individual symbols and can be divided into upper and lower parts. Here, basic rules refer to *AND*, *OR*, and *XOR* rules. More sophisticated rules refer to *COMPLEX* rules and *EVENT-BASED* rules. While the first kind groups a combination of basic rules, the latter combines basic rules and events.

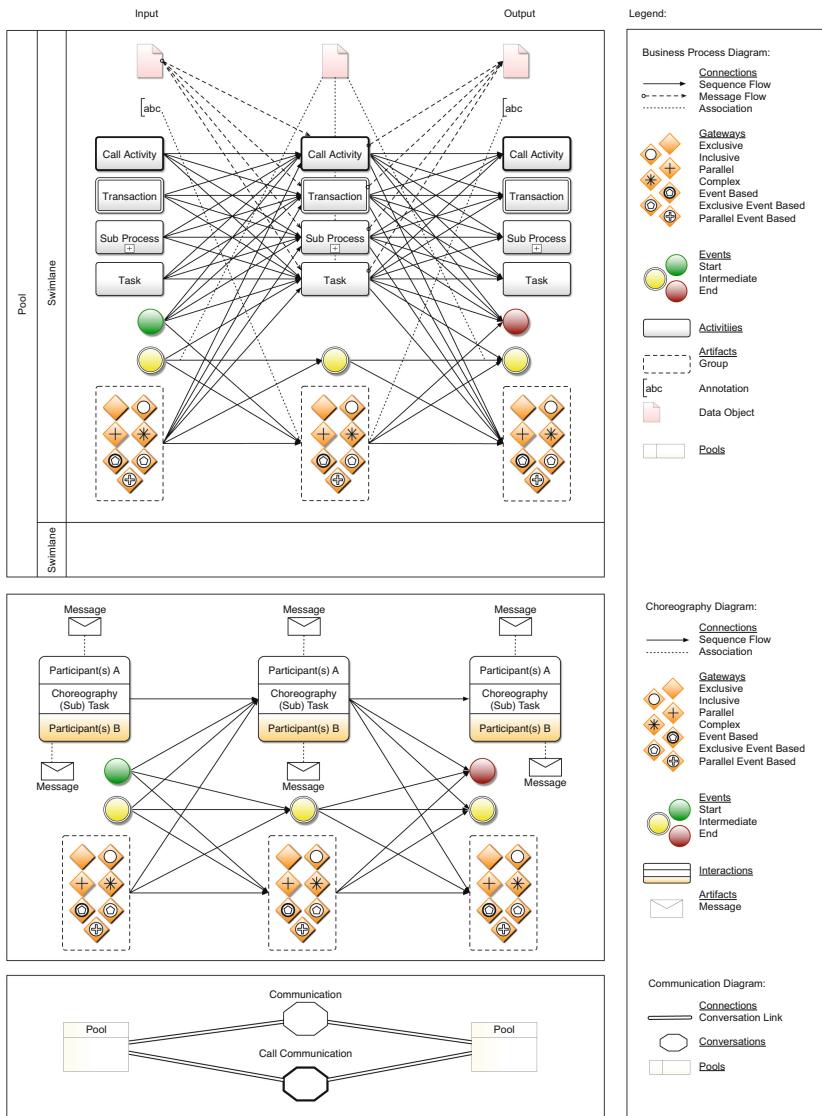


Figure 2.15 Generic BPMN business process meta-model (synthesized from literature)

7. Edges connect two activities corresponding to their sequence of action so that they are directed.
8. Edges and activities can be associated with artifacts, such as *annotations* or *data objects*.
9. A *role* or *organizational entity* realizing a task is visualized on behalf of *swim-lanes* being part of an organization *pool*.
10. All kinds of activities can demand for and produce *data objects*, which are connected by incoming and outgoing *message flows*.
11. *Annotations* are associated with their corresponding edge.

As the BPMN is rather semi-formal than strictly formally defined, the presented guidelines must be seen as recommendations and might be extended with easily interpretable rules by the model constructor, such as message flows among swim-lanes or boundaries of pools, individual pictographs, associations, etc. This set of rules is extended by design rules for choreography diagrams (Object Management Group, 2013):

12. The diagram's core node refers to *interactions*, *events*, and *control nodes*.
13. The interaction's name should reflect its time-consuming perspective of the message exchanges.
14. BPMN-based choreography models are initiated with a *start event* and finish with an *end event*.
15. Within the diagrams, *interim events* can affect the interactions.
16. Each diagram at least consists of one type of *interaction*.
17. Control nodes are represented by rectangles as well as their individual symbols and can be divided into upper and lower parts. Here, basic rules refer to *AND*, *OR*, and *XOR* rules. More sophisticated rules refer to *COMPLEX* rules and *EVENT-BASED* rules. While the first kind groups a combination of basic rules, the latter group combines basic rules and events.
18. Edges connect two interactions corresponding to their sequence of message exchanges so that they are directed.
19. Interactions can be associated with artifacts, such as *message objects*, so that artifacts are associated with their active participant.

Rules are further extended by design rules for communication diagrams (Object Management Group, 2013):

20. The diagram's core node refers to *pools* and *conversations*.
21. Conversations can refer to either *communications* or *call communications*.

22. The conversation's name should reflect its cluster of message flows.
23. Process participants are visualized by *pools*.
24. Edges connect two process participants corresponding to their joint message flow. Since this includes messages in both directions, edges are not directed.
25. Each diagram at least consists of one *conversation* and two *participants*.

Example: An example model for a BPMN is visualized in Fig. 2.16. It shows a process for unsolicited applications with the help of the three BPMN diagram types and thus clarifies the application of BPMN conventions.

In the figure on the top, one can see the BPD. It shows the collaboration of three roles, which are the *applicant*, the *staff department*, and the *operational departments*. Among them, *data objects* are distributed as *message flows* defined. From the applicant's perspective, objects refer to the *application* document, a *resubmission request* and an email having either an *invitation* or *friendly rejection*. While the *staff department* is responsible for the document completion and applicant communication, operational departments check applications for requirement fulfillment.

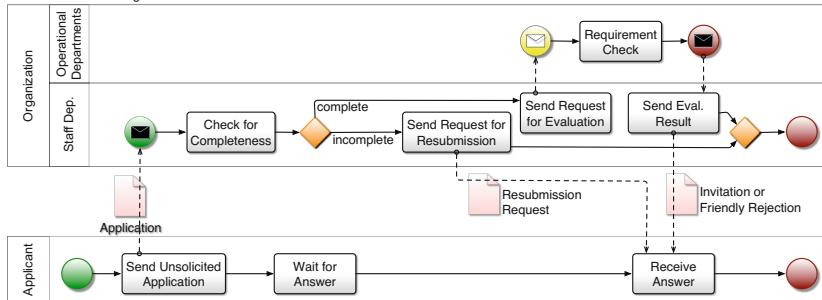
In the center of the figure, one can see the same process modeled with the help of the choreography. While this shows the same constellation of control flows, gates and events, a focus lies on the interaction of at least two process participants.

On the bottom of the figure, the communication diagram visualizes the communication among process participants. It is apparent that the applicant and the staff department as well as the staff department and operational departments are communicating. Thus, operational departments are not in touch with applicants during the application completion process.

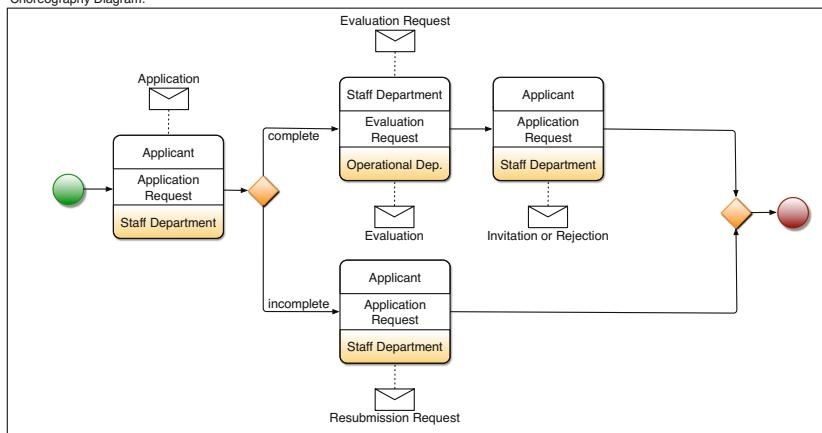
Based on this figure, one can identify the effective integration of several process participants on behalf of the process modeling from the perspective of each participant at the BPD. As message flows even define start and end events for participants, process implementations on base of workflows are well prepared and can be realized by workflow management systems easily. Further, the abstraction of the BPD is apparent from the choreography diagram and communication diagram. Since the latter two diagram types can be derived from the BPD, and the CoNM requires the direct modeling of detailed processes, the following focuses on the BPD only.

Critical appraisal: The BPMN has its strengths in terms of modeling various process participant perspectives and their coordination. As modeling items are provided for common business process contexts, and further extensive variations of them are provided, there is an item for almost every situation. Common modeling language criteria from the business process domain (including function and behavioral perspectives) can be satisfied. Unfortunately, this flexibility is realized at the cost of

Business Process Diagram:



Choreography Diagram:



Communication Diagram:

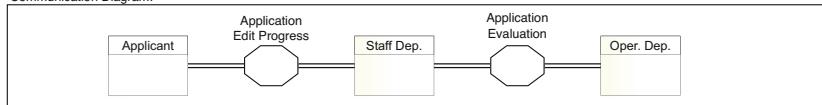


Figure 2.16 BPMN-based process model example (created to show the strength of the object of investigation)

decreased interpretability because nobody is aware of all items and individual design conventions. Further, neither a positioning of processes within the real world nor their augmentation is considered by the BPMN.

Since a common architecture, such as the house of ARIS (see section 2.6.1.2) is not considered by the BPMN, integrative modeling is not supported. The mapping of organizational entities and roles can only be realized by using the modeling items of pools and lanes. The integration with an organization model, which is not redundant because of an underlying data base concept is not enabled. In addition, data models and functional breakdowns are beyond the scope of BPMN. Further, the capability to present overviews of the company's entire processes including their link to business objects and message flows is not enabled as well. Thus, overall, only concepts of modeling applicable to business processes are enabled, which does not consider 3D models and augmentations.

An adequate process modeling for the knowledge management domain is not realizable since information flows and knowledge flows including their context items are not provided. Thus, the corresponding modeling language criteria are not satisfied at all. Further, the simulation context is not considered and so modeling language criteria from the simulation domain are dissatisfied. As the BPMN's syntax is semi-formal and extremely flexible, the implementation in modeling tools is difficult. Since simulation-specific parameters are not considered by the BPMN at all, the simulation of BPD particularly needs circumstantial extensions for the CoNM construction. Since neither modeling items nor ANN techniques are considered by the BPMN, they do not satisfy modeling items from the ANN domain and ANN tool criteria.

Since the BPMN provides neither a focus on knowledge transfers nor on ANN, it is only suitable for the context of business processes. Thus, it is not suited for a combination with ANN tools as a basis for the construction of a CoNM.

2.1.3.5 Critical Appraisal of PMLs

Considering the aforementioned process modeling approaches and its interim conclusions, it seems that today's (business) process modeling approaches show numerous aspects, which can be found in process modeling taxonomies (section 2.1.2). However, they are still in development stage and do not consider ANN techniques:

- None of the presented process modeling languages consider neuronal processes, neuronal modeling objects and therefore, an integration of ANN objects with currently available process modeling perspectives is missing.
- Only some process modeling language approaches consider an *EA concept* (for e.g., ARIS), which supports the construction of consistent process models being

integrated perfectly within a company's structure. This is essential as workflows can be optimized by neuronal mechanisms.

- None of the presented process modeling language approaches considers phases dealing with neuronal libraries, neuronal tools, neuronal methods, or neuronal concepts. An integration with currently available concepts, tools, or libraries is therefore missing as well.

A research gap becomes evident here.

A synthesis of any process modeling approach presented here, enriched with ANN modeling objects and required concepts and mechanisms, will support a broad acceptance of the CoNM in corresponding research communities. Further, a synthesis will ensure its extensive application possibilities.

The idea for a 'holistic' synthesis efficiently representing all modeling perspectives as well as objects and being applicable in all modeling situations is not new (Kueng et al., 1996, p.98; Giaglis, 2001, p.14; Sultanow et al., 2012, p.3310). Moreover, experiences show that the multiplicity of possible modeling goals and objectives lead to impractical modeling approaches. Therefore, a synthesis must ensure that complex models do not reduce the ease of use for any realistic setting (Curtis, Kellner, and Over, 1992).

Hence, for the construction of a CoNM, a compromise between the holism of modeling items required and the practicability must be identified . In principle, a practical and holistic synthesis must address the following aspects:

- It must broadly consider business process-relevant objects and consider their operational behavior with regard to managerial co-ordination, control, development, and policy (Beer, 1979).
- According to Kueng et al. (1996, p. 98), it must support the process of modeling. Here, the kind of information that is required as input for the methodology and output from its application must be made clear.
- Inspired by Fröming (2009, p.3310) and Sultanow et al. (2012, p.3310), all knowledge and knowledge flow-oriented issues must be addressed.
- According to Scheer (2002, p. 112), it must not be limited to the design of processes but consider tools, methods, and architectural components.
- ANN techniques must be harmonized with the understanding of the business process modeling and knowledge modeling.

The creation of a CoNM will therefore consider the aforementioned points and it will be based on a practical synthesis of process modeling languages presented here.

2.1.4 Conclusion About Taxonomies and PMLs

The conclusion about process modeling is based on the following. First, taxonomic approaches are appraised. Then, process modeling languages (PMLs) follow. A process modeling-specific facet is subsequently identified.

Taxonomy: Most approaches to classify process modeling notation components, known as taxonomies, are based on a set of *perspectives*, *approaches*, or *architectures*. The consideration of neuronal perspectives, neuronal modeling objects and an integration with components from a process modeling domain have not been identified in any approach (see each individual interim conclusion). Since taxonomies were defined to classify attractive process modeling elements, which was not aimed at the construction or realization of a CoNM, an appropriate CoNM taxonomy is still missing essentially. Hence, taxonomies neither have been evaluated as the foundation for the CoNM (for e.g., as building blocks on various levels) nor have they been evaluated to classify elements required by a CoNM so far.

In order to decide whether current taxonomy can at least partly identify a CoNM foundation or which taxonomy criteria are suited to be part of a CoNM criteria catalog, one has to evaluate the taxonomic approaches: Since the CoNM is intended to be carried out in a holistic and general setting, any criteria provided by *perspective-oriented taxonomies* is relevant as perspectives focus on aspects that are accepted in the process modeling community. The same counts for *approach-oriented taxonomies* because approach elements focus on aspects that are accepted in the process modeling community. Moreover, the same counts for the *architectures* presented since architectural building blocks focus on aspects that are accepted in the process modeling community. When applying the accepted concepts from the process domain in the ANN process domain, all of their criteria are highly relevant.

Since none of the mentioned approaches provided the criteria for the use of neuronal techniques and concepts, which is an essential part of the ANN process domain, none of the evaluation frameworks can at least partly identify a CoNM foundation. Further, this implies that the process modeling domain has not been brought together with AI techniques yet. A research gap becomes apparent here, which requires the following: first, the construction of a CoNM because anything similar is missing so far; second, a CoNM taxonomy selection system that extends

common criteria and can evaluate current taxonomies; third, an analysis of currently available taxonomies for the use of CoNM-relevant aspects, such as neuronal techniques, specific modeling objects, and common tool criteria.

Process Modeling Languages: Most approaches for process modeling language are based on individual sets of *modeling items* and *syntax*, each having a unique modeling strength. The consideration of neuronal perspectives, neuronal modeling objects, and an integration with components from a process modeling domain have not been identified in any approach (see each individual critical appraisal). Since process modeling languages were defined to carry out process modeling concepts which do not refer to models coming from a CoNM, an appropriate CoNM process modeling language is still missing per se. Hence, process modeling language has neither been evaluated to be the foundation for the CoNM (for instance, by providing commonly accepted modeling items and well-defined syntax) nor have process modeling languages been evaluated as part of a CoNM so far.

In order to decide whether current process modeling languages can at least partly be used as a CoNM foundation, or which criteria are suited to be part of a CoNM evaluation criteria catalog, the following aspects are considered category-wise:

1. Since the CoNM is intended to be carried out in a holistic and general setting, any *modeling item* provided by process modeling languages is relevant since items focus on aspects which are accepted in the process modeling community. When applying the accepted concepts from the process domain in the ANN process domain, these criteria are highly relevant.
2. Since the CoNM is intended to be carried out in a holistic and general setting, any *syntax or convention* provided by process modeling languages is relevant as syntax and convention focus on aspects which are accepted in the process modeling community. When applying the accepted concepts from the process domain in the ANN process domain, these criteria are highly relevant.
3. Since the CoNM is intended to be carried out in a holistic and general setting, any *modeling concept* being considered by process modeling languages is relevant since modeling concepts focus on aspects which are accepted in the process modeling community. When applying the accepted concepts from the process domain in the ANN process domain, those criteria are highly relevant.

Since no process modeling language provided items or syntax about the use of neuronal techniques and concepts which is an essential part of the ANN process domain, none of the provided process modeling languages can serve as the modeling language of the CoNM. Further, this implies that the process modeling domain

has not been brought together with AI techniques yet. A research gap becomes visible here which requires the following: first, the construction of a CoNM process modeling language because anything similar is missing so far; second, a CoNM process modeling language selection system that extends common criteria and is able to evaluate current process modeling languages to be part of a CoNM foundation; third, an analysis of currently available process modeling languages for the use of CoNM relevant aspects, such as neuronal techniques, specific modeling objects, and common tool criteria.

Facit: Neither process taxonomies and taxonomy evaluation frameworks consider the integration of the process domain and the ANN domain nor process modeling languages do so. Criteria collections do not provide ANN and Deep Learning criteria. Further, process modeling languages do not realize the aforementioned criteria. Thus, a research gap within the process domain becomes visible here. This requires the following: first, the construction of a CoNM taxonomy; second, the design and demonstration of a CoNM selection and evaluation framework; third, the implementation and demonstration of a CoNM process modeling language.

2.2 Knowledge Modeling

While process modeling focuses on the structural and behavioral modeling of processes (section 2.1), knowledge modeling attempts to incorporate the context of knowledge by the provision of structural knowledge elements as well as the consideration of their dynamic effects. Since knowledge evolves during the execution of an activity, knowledge must be considered structural as well as having processual effects, and the terms *knowledge* and *knowing* are equivalent (Neuweg, 1999, p. 135; Renzl, 2004, p. 38; Gronau, 2009, p. 6). Therefore, ideally, common approaches to represent processes are combined with knowledge modeling approaches so that the behavioral side of knowledge can be recognized from their processual perspective as well. Accordingly, a potential can be identified in the recognition of knowledge within the processual behavior of complex structures such as ANN. The underlying hypothesis of this contribution is that insights into the working of ANN can be realized by the recognition of processual knowledge entities within neuronal structures and thus the interpretability of ANN can be increased and, for example, made accessible for process optimizations.

The knowledge modeling context is characterized by the following: In a first sub-section, definitions for and about basic concepts are established. The second sub-section provides concepts concerning the management of knowledge so that

the following questions about the processual organization of knowledge management can be addressed: Are common economical principles organized similar to biological structures of the human brain? Can organization structures be organized in a way similar to the human brain which can process optimized and efficient concepts through evolution over centuries? Do management principles already consider mechanisms of neuronal networks or can machine learning approaches of ANN be optimized by the use of economic principles? In this case, there can be a potential application of concepts of one domain in the other domain and vice versa. Thereafter, in the third sub-section, current standards about knowledge modeling languages are examined with respect to the use of ANN-required concepts. In a fourth sub-section, a domain-specific research need is identified.

2.2.1 Basic Concepts and Definitions

Basic concepts refer to the definition of knowledge and an understanding of the concept of learning, which has been considered by various disciplines (section 2.2.1.1). Different forms and the representation formalisms of knowledge are discussed thereafter in section 2.2.1.2 so that the progress of knowledge model creation, what is called knowledge modeling, is prepared. Section 2.2.1.3 then presents the definition of knowledge generation and transfer processes as sub-set of processes (presented in section 2.1).

All combined, this stands as a foundation for the following two kinds of investigations. First is for the examination of knowledge management concepts as OoIs since these can be interpreted as higher order processes and consider the management of knowledge and learning organizations. Second is for the examination of knowledge modeling languages (as further OoIs) to reflect the processual understanding of knowledge. Further, the provided concepts serve as building blocks for the construction of a CoNM.

2.2.1.1 Learning and Knowledge

A literature search of the term “knowledge” identified that the concept of *knowledge* as a construct of various disciplines has numerous definitions, each requiring its corresponding understanding of “learning”. *Inter alia*, attempts for definitions can be found in philosophy, the domain of knowledge management, information system theory, educational contexts, everyday understanding, and in many more areas. As the CoNM intends to integrate the process domain and ANN domain and the identification of knowledge within neuronal structures is intended, an adequate definition of knowledge must bring together aspects from a non-machine side inspired by the

process domain (everyday, philosophical, pedagogical, and knowledge management contexts, and so on) as well as aspects from a machine side inspired by the ANN domain (information system, machine learning contexts). This is so that a definition supports everyday understanding about knowledge-related concepts and can serve as a foundation for commonly known concepts.

The following presents attempts to define knowledge and the corresponding understanding of cluster-wise learning according to Grum (2020). Although clusters must be seen on an equal footing, the presentation starts with everyday-based understandings as a kind of introduction to the definition of knowledge. Philosophical paths start in ancient times and still influence the modern understanding of knowledge-related disciplines, which therefore is presented as the second cluster. Abstract definitions are presented subsequently, which refer to educational, pedagogical, and cognitive-psychological clusters as well as definitions from knowledge management (KM, Schmitz and Zucker, 2003; Gronau, 2009) and knowledge logistic contexts (KL, Hartlieb, 2002).

The subsequent two clusters focus on the machine side so that information science (IS)-based and artificial intelligence (AI)-based definitions as well as neuro-scientific and cognitive-scientific definitions are presented. With this, approaches are presented from abstract to detailed, from informal to formal, and from ancient to modern perspectives. The critical appraisal section then presents knowledge definitions and learning understandings in the context of CoNM construction. Hence, the interim conclusion section establishes a well-founded definition of knowledge, which is adequate for CoNM construction.

2.2.1.1 Definitions Based on Everyday Understanding

In everyday contexts, knowledge is essentially interpreted as awareness about something (Eberhard, 1802; Scheeben, 1875; Rahner and Vorgrimler, 1969; Colburn, 2000; Reichelt, 2014; Skwara, 2016; Schlabach, 2018; Skwara, 2018). Only seldom is it differentiated into *knowledge* and *information*. Who is “informed” about something, “knows” something and who “knows” something is able to transfer “information” (Rothman et al., 2009). This understanding can be related to the linguistic meaning of the term “knowledge”: the Indo-European origin of this term means “I have seen” or “I know”, from which the Latin word “videre” (seeing) and the Sanskrit word “veda” (knowledge) evolved (Skwara, 2016; Skwara, 2018).

Some definitions present knowledge as an everyday term without a clear meaning: As soon as an example knowledge term is used with different understandings, for e.g., because of a different situational context, clearly and strictly formalized taxonomic classifications of the same terms do not reflect the same meaning. A com-

mon understanding is rather achieved because of *family resemblances* (Wittgenstein, 1953, § 67; Prechtl and Burkard, 2015). Here, each knowledge term is associated with various attributes so that knowledge terms can be reflected in various contexts and taxonomic classifications because of their similarity.

Based on everyday life, learning refers to the acquisition of information. If someone does not know something and then learns it, they acquire the corresponding information in terms of knowledge.

2.2.1.1.2 Philosophical Definitions

Under the philosophical branch known as *epistemology* or *theory of knowledge*, philosophers studied the nature of knowledge, justification, and the rationality of belief over centuries, beginning from ancient times. They defined conditions for awareness, the generation of knowledge, and its differentiation of other forms, such as beliefs and convictions. The philosophical term of *knowledge* is commonly interpreted to be a human awareness of facts (similar to the everyday understanding described in section 2.2.1.1.1). This differentiates the terms *meaning* and *believing* based on certain rational reasons and justifications (Brendel, 2013). Fig. 2.17 shows philosophical movements defining knowledge as well as branches or philosophical schools providing interpretations. Main definitions are explained in the following section. As not all definitions can precisely be mapped to their philosophical schools (Reinmann-Rothmeier and Mandl, 2000), each school follows its own interpretations of knowledge conceptions (Ichikawa and Steup, 2018a).

Traditional conceptions: The initial definitions about philosophical knowledge conceptions were formulated by ancient schools. When Plato stated the existence of *apriori knowledge*, which is not obtained by the perception of someone's senses but deductively through logical thinking (cf. *Menon* 97e–98a; *Theaitet* 202b–c; *Politeia* 509d ff, 477b–e; *Gorgias* 454d), the philosophical school of *rationalism* was founded. Probably, its most prominent scholarly representative was René Descartes.

Aristotle argued the existence of a priori knowledge as a discourse. According to him, knowledge only could be obtained through perception from senses and the inductive reasoning of the same. (Aristot. an. post. 72b, an. pr. 67 a1 ff, an. post. 73 a22, an. post. 71 b1 ff, an. pr. 67 b1 ff; an. 3.4, 429a10 ff). Thus, the philosophical school of *empiricism* was founded whose most prominent scholarly representative was probably John Locke.

More modern schools attempted to bring together the empirical and rationalistic perspectives to combine inductive and deductive approaches to gain knowl-

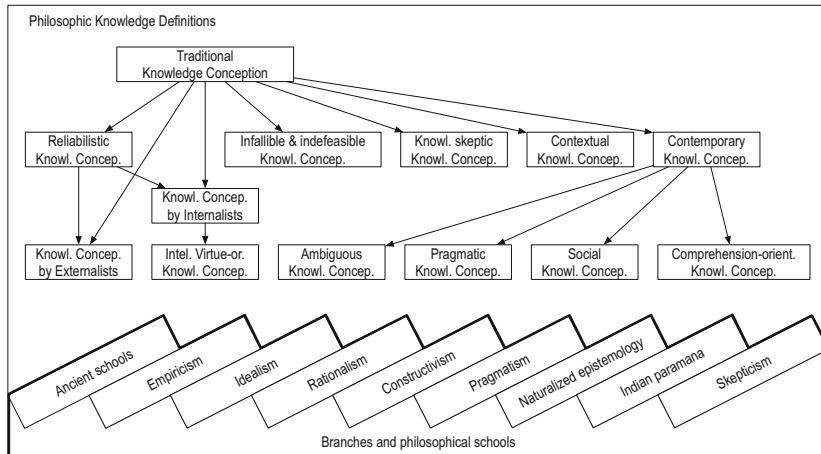


Figure 2.17 Philosophical streams to define knowledge (synthesized from literature)

edge. Prominent scholarly representatives are Kant, Hegel, and Diderot (Prechtl and Burkard, 2008, p. 683–684).

All these conceptions are commonly subsumed as *traditional knowledge conceptions* (Brendel, 2013). Accordingly, knowledge can be defined as a *true and justified opinion*. As ordinary guessing and wishful thinking does not lead to knowledge, even if the true opinion is identified by random events, knowledge is only identified if it is justified by argumentation. So far, the definition of knowledge has been found to be associated with conditions that are individually necessary and jointly sufficient for propositional knowledge, thoroughly answering the question what it takes to know something. Commonly, the first condition is referred to as *truth condition*, the second *belief condition*, and the third *justification condition* (Ichikawa and Steup, 2018b). Considering the requirement for having true and justified opinions that one would not qualify as knowledge (Gettier, 1963), Gettier's contradictory examples lead to the establishment of various extensions and alternative knowledge conceptions. As some of them are mutually exclusive, they overrule previous conditions or introduce additional conditions as the following demonstrates.

Reliabilistic conceptions: *Reliabilistic knowledge conceptions* define knowledge as a *true opinion* built on the basis of a *robust opinion building process* (Goldman, 1986; Goldman and Beddor, 2016). For instance, Armstrong introduced a fourth

condition, that the opinion must not be built on the basis of wrong assumptions for it to be characterized as knowledge (Armstrong, 1973). Additional conditions were presented by Lehrer and Jr (1969) and Goldman (1976). Particularly, Goldman argued against true assumptions being sufficient and replaced the justification condition with a condition about causal reliability. However, approaches here struggle to define robust opinion-building processes on the level of understanding or any further level (known as *generality problem* by Feldman, 1985; Goldman, 1986).

Sub-groups further differentiate between *internalistic* and *externalistic* knowledge conceptions: While both groups consider knowledge a *true opinion*, the first group requires the *subject to be able* to justify its opinion by cognitively accessible and rational reasons. Modern approaches here particularly address the availability of intellectual capabilities and rational virtues to obtain knowledge (DePaul and Zagzebski, 2003), which is referred to as *intellectual virtue-oriented* knowledge conceptions (Brendel, 2013). The latter group does not require the subject to justify its opinion. Here, third parties such as servants are allowed to justify the subject's knowledge. So, *inter alia*, this allows animals to obtain knowledge as well.

Further conceptions: Other philosophical approaches require the justification of knowledge to be *infallible* (Kirkham, 1984) and demand that there should be *no overriding or defeating truths* for the reasons that justify one's belief. This is referred to as an alternative fourth condition ("condition of indefeasibility", Matilal, 1986). *Knowledge skeptic* conceptions defeat the possibility to potentially define knowledge at all as it is impossible to eliminate the idea that all perceptions can be fooled systematically by a higher instance. Thus, *contextual* knowledge conceptions defeat a substantial understanding of knowledge through the introduction of a context (Brendel and Jäger, 2005): If any knowledge statement is considered with its specific context, it is valid for exactly this context. Other conceptions focus on modal conditions connected with the sensitivity (knowledge must stand any circumstances), and safety (knowledge must stand counterfactual terms) as well as the requirement that knowledge must rule out relevant alternatives. They question the analyzability of knowledge at all (Ichikawa and Steup, 2018b).

Contemporary conceptions: Contemporary philosophical knowledge conceptions change the focus of knowledge definitions. *Ambiguous knowledge conceptions* question the possibility to establish one unique definition because of the assumption that knowledge has several meanings and must satisfy definitions from various domains (Ernst, 2002). *Pragmatic knowledge conceptions* radically neglect the attempt to analyze the term knowledge and address *purpose-related* knowledge characteristics (Craig and Vossenkuhl, 1993). *Social dimensions* of knowledge

acquisition and knowledge transfer are discussed in *social knowledge conceptions* (Goldman, 1999). Others draw attention to *comprehension* rather than knowledge (Brendel, 2013) and are referred to as *comprehension-oriented knowledge conceptions*.

Learning: Independent from the specific philosophical approach to define knowledge, learning in a philosophical context can be interpreted as an opinion-building process, fulfilling more or less definition-specific conditions.

2.2.1.1.3 Educational Definitions

Although definitions coming from pedagogical contexts and definitions from an educational context are closely related, the latter rather presents knowledge definitions focusing on *competences*. So, knowledge is defined to be a pedagogical concept of enabling individuals to act in specific situations, which is referred to as competence (Gronau and Grum, 2019). In the literature, one can find various kinds of competences: personal, communicative, professional, methodological, activity and action, personal, personality, soft skills, and so on. Competence models even build on Heyse and Erpenbeck's competence atlas listing 64 competence areas (Heyse and Erpenbeck, 2009; Mietzner and Kamprath, 2013).

Weinert (2001) defines competences as *cognitive abilities and skills of individuals*, which are available or learned in order to *solve certain problems*, along with their connected motivational, volitional, and social willingness and abilities to create and *implement problem solutions in variable situations* responsibly. This includes all personal conditions required for the execution of a certain activity, such as the creation of a problem solution. It also includes skills in the form of learned or acquired behavioral patterns, such as experience, the use of knowledge, or the level of practice (Gronau and Grum, 2019). Hence, the learning refers to the acquisition of new skills and requires that a certain set of abilities and skills be mastered on less demanding cognitive levels. The reference to *various situations* concerns the availability of different levels of validity in compliance with different *contexts*.

According to Bloom et al. (1956), competencies can be categorized in six levels: 1) knowledge, 2) comprehension, 3) appliance, 4) analysis, 5) synthesis, and 6) evaluation. Since knowledge is considered here as the first competency level, developing levels require the use of more sophisticated competences so that higher levels of knowledge are constructed.

Learning: Contrary to psychological definitions about knowledge (introduced in section 2.2.1.1.4), competence-related knowledge definitions consider a purpose-

oriented use regarding a problem as well as its situational implementation. Thus, learning refers to the achievement of higher levels of competencies and learning success can be measured by tests focusing on the intended level of competencies.

2.2.1.1.4 Pedagogical and Cognitive-Psychological Definitions

Definitions from pedagogical contexts build on psychological models and define knowledge as *information* being perceived by a person and integrated in the person's *long-term memory* in form of a semantic representation and having a *schemata* structure (Schmidt, 2000, p. 90; Wellenreuther, 2012, p. 9, 13, 17). An example for a schemata can be found in the identification of an animal. Although no cat looks like another cat, the corresponding schemata of a cat makes us react in the same way whenever we see a cat. As *atomic* schemata can be combined with more *complex* schemata, one, therefore, considers a *knowledge structure* (Wellenreuther, 2012, p. 9, 17). An example for a schemata combination can be found in the act of reading. While reading, basic schemata refers to the identification of individual letters. More complex schemata refers to the identification of their combination into words, words into sentences, and sentences into thoughts.

Contrary to the understanding of the term "information" in KM domains (introduced in section 2.2.1.1.5), here, *information* refers to the perception of sensory data, for e.g., visual data perceived by the eyes or acoustic data perceived by ears. The denomination as *information* might be associated with higher mental activities, such as *drawing attention* and establishing a *situational interpretation*, which all might enrich simple *data* with *information*.

Learning: As information needs to be cognitively prepared for inclusion in the long-term memory, a learning considers two phases (Wellenreuther, 2012, p. 9), each demanding a pedagogical valuable processing and individual understanding of learning. Hence, the different meanings of an understanding of learning are established by each phase individually as follows.

First learning phase: The first learning phase focuses on the acquisition of new knowledge in the *working memory* and its transfer to the long-term memory (Baddeley, 1986; Baddeley, Gathercole, and Papagno, 1998; Gathercole, 1998). It begins with the perception of information and is not finalized until the very first schemata is built in the long-term memory. Fig. 2.18 presents this process for both kinds of memories.

On the left of the visualization, one can see the information processing from perception to the storing of schemata in long-term memory. Here, four kinds of

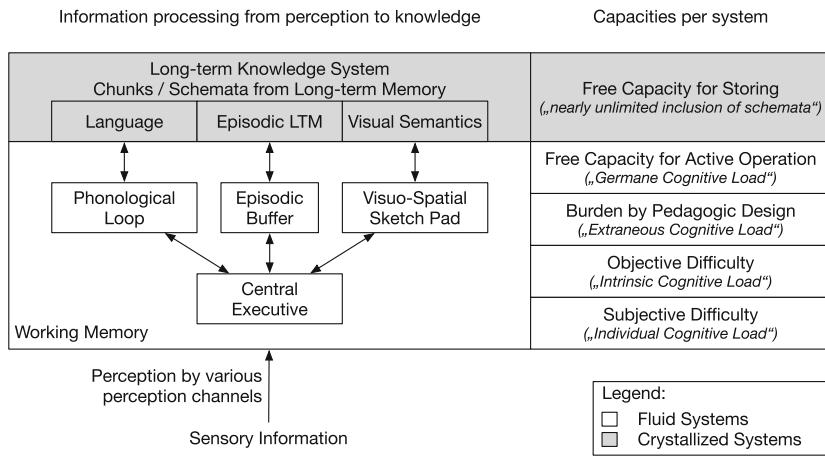


Figure 2.18 Memory and overload theory (synthesized from Baddeley, 1986; Baddeley, Gathercole, and Papagno, 1998; Gathercole, 1998)

systems can be found (Baddeley, 2002; Baddeley, 2003). First is the *phonological loop*, which is responsible for storing phonological information referring to the sound of language. By continuously refreshing it in a rehearsal loop, the system prevents it from being forgotten. Second is the *visuo-spatial sketch pad*, which is responsible for storing visual and spatial information, for e.g., for the construction of and perception of visual images or mental maps. Third is the *episodic buffer*, which is responsible for holding representations that integrate phonological, visual, and spatial information. Further, it holds information not covered by the slave systems (Baddeley, 2000; Baddeley, 2012). Fourth is the *central executive*, which directs attention to relevant information, suppresses irrelevant information and inappropriate actions. Further, it coordinates cognitive processes when more than one task is simultaneously performed so that it functions as a master system for the three slave systems mentioned previously. Based on the distinction of these systems, one might argue exactly when knowledge is identified. For example, everyday-based knowledge conceptions even identify knowledge when it is reported on the basis of the working memory although it has not been anchored in the long-term memory yet. In contrast, the knowledge management domain identifies knowledge not when it can be accessed from the long-term memory. Since the pedagogical and cognitive-psychological knowledge conceptions focus on *schemata*, knowledge is considered

from the point of the perception via perception channels in the working memory up to the point where it has been stored in the long-term memory. Therefore, its manipulation during this entire path is essential and an understanding of learning is established specifically on the basis of the following burdens.

Burdens of the first learning phase: Generally, the working memory systems can deal with about seven *chunks* referring to small entities of information, which can differ in size, form, and structure (Sweller, Merrienboer, and Paas, 1998). As the *capacity* of the working memory to process these chunks is further limited by burdens, its systems functions as a bottle neck for learning processes. On the right of Fig. 2.18, one can see a collection of four burdens limiting the capacity in accordance with the cognitive load theory (Sweller, Merrienboer, and Paas, 1998; Clark et al., 2006). The first burden describes a *subjective difficulty*, which can refer to lack of prior knowledge and personal factors, such as worries, fears, or priorities. The second kind of burdens is called *objective difficulties* and refers to learning tasks requiring too many chunks. Here, task-relevant knowledge can be broken down so that some schemata are accessible via the long-term memory. The third kind of burdens describe the preparation of learning materials required for the learning tasks (*burdens by pedagogical design*). This refers, for instance, to the design of training material, its texts, graphics, and the use of examples. Fourth, missing *free capacities for active operation* can hinder the learning process. If capacities required for learning relevant activities are not available, for e.g., because capacities are bound in complicated scholarly discussion, the learning process cannot be realized successfully. Since the capacity of the working memory is limited, and the schemata currently built are lost if not stabilized by the second learning phase, these needs must be addressed by a pedagogical valuable learning. Therefore, during the first learning phase, the understanding of learning refers to the effective schemata building process, which is in harmony with learning burdens and the available capacities and is best supported by pedagogical valuable learning methods.

Second learning phase: The second learning phase focuses on the stabilization of learned knowledge in the long-term memory. In this case, methods such as *repetition* and *practicing* support the linking of new schemata with the current knowledge base so that an improved knowledge structure is realized. Only then, schemata are accessible for a long-period of time, either in a cognizant or a tacit manner (Heymann, 1998, p. 7). As knowledge can only be stored temporarily in the working memory, *fluid* systems are considered that require only temporary activation. Further, the *crystallized* system of the long-term memory comes into play. Following the common distinction of *working memory* and *long-term memory*, from a pedagogical

perspective, the teacher has to adjust the amount of information to be learned to the biological circumstances of the individual's *working memory* in the first learning phase. This includes dealing with capacity issues (cognitive-load theory following Sweller, Merriënboer, and Paas, 1998; Clark et al., 2006) and emotional factors of individual learners (Lufi, Okasha, and Cohen, 2004), as well as the sustainable anchoring of schemata in the *long-term memory* in the second learning phase. Thus, during the second learning phase, the understanding of learning refers to effective schemata anchoring, which is best supported by pedagogical valuable learning methods but is different from the learning method use of the first learning phase.

In conclusion, while knowledge in the first learning phase refers to simple schemata within the working memory, knowledge rather refers to a complex system of schemata within the long-term knowledge system in the second learning phase. Regarding the current focus on a learning phase, each phase requires an individual understanding of learning and follows different learning objectives.

2.2.1.1.5 KM- and KL-Oriented Definitions

The domain of knowledge management (KM, Schmitz and Zucker, 2003; Gronau, 2009) and knowledge logistic (KL, Hartlieb, 2002) define knowledge as the unity of skills, cognition, and capabilities used by knowledge carriers for the solution of given problems (Davenport and Prusak, 2000; Krogh, Ichijo, and Nonaka, 2000; Polanyi and Sen, 2009). Some consider knowledge as humanized information (Telesko, Karagiannis, and Woitsch, 2001; Karagiannis and Woitsch, 2002, p. 38). The following aspects are essential.

Contextual Meaning: Generally, knowledge definitions associate knowledge with its use with respect to specific organizational objectives. This justifies why knowledge is considered on a rather fuzzy abstraction level that allows its management than on absolute truth or hard facts (Krogh, Ichijo, and Nonaka, 2000, p. 7). Based on an organization-wide knowledge base, the question of how data, information, and knowledge are managed can be addressed such that an individual can solve organizational tasks or problems efficiently. Therefore, knowledge is considered with the contextual meaning of its problem to be solved and with its organizational objectives to be satisfied.

Dynamic Nature: Further definitions propose that knowledge changes through reflections about it and evolves over time. With the need to formalize knowledge as a basis for IS supporting its management, it is considered a temporary true state variable (static knowledge understanding) and a self-reference process (dynamic knowl-

edge understanding) from the perspective of knowledge management (Schmitz and Zucker, 2003; Gronau, 2009) as well as from the KL context (Hartlieb, 2002). While the first allows the consideration of knowledge as an object which can be mapped in and managed with help of IS, the second must consider knowledge as a *processual phenomenon* or a *complex system* that can *be managed* by an adequate environment (Allee, 1997; Gronau, 2009).

Tacit Dimension: Required for an adequate organizational knowledge management, the following terms are differentiated by definitions: First is *knowledge*, which is considered as information enriched with contextual experience. It is further differentiated by tacit and explicit knowledge requiring individual management attempts (Reber, 1989; Nonaka and Takeuchi, 1995; Smith, 2001; Gronau, 2009). *Tacit knowledge* cannot be documented without significant effort. Beside motor and sensory abilities, this includes personal experiences and is related to familiarity with procedures and the evaluation of situational (contextual) circumstances. *Explicit knowledge* can be described, documented, and communicated. Second is *information* or data which makes a difference for the receiver when perceived (Colburn, 2000). Thus, data, for instance, leads to the solution of a given problem when perceived. Contrary to the pedagogical information understanding, here, the focus lies on the effect of information because of its perception, which excludes various knowledge schemata identified by pedagogical understanding. Third is *data* as information stored at machines (Colburn, 2000) and thus refers to a set of values about subjects with respect to qualitative or quantitative variables.

Social Construction: Some knowledge definitions focus on the activity of a social construction (Gronau, 2009). As the knowledge carrier is considered in its individual organizational context, the knowledge construction is realized within its individual knowledge network considering socializations on various ontological levels (Nonaka and Takeuchi, 1995). Examples can be found in knowledge being constructed by individuals, groups, departments, and other organizational entities. Thus, knowledge can either be directly associated with individual knowledge carriers or it can consider social systems to be knowledge carriers so that one can differentiate between individually available and socially accessible forms of knowledge (Reinmann-Rothmeier and Mandl, 2000).

Learning: Independent from the specific KM- and KL-specific approach to define knowledge, in terms of KM and KL, learning refers to the enrichment of skills, cognition and capabilities of either individuals, groups, departments and organiza-

tional entities, as well as the improved use of knowledge in the sense of *learning organizations* (these will be introduced in section 2.4.3).

2.2.1.1.6 IS-Oriented and AI-Based Definitions

Information system oriented knowledge conceptions define knowledge as *information* which is valuable in specific situations and supports the identification of decisions (Lämmel and Cleve, 2012). Thus, knowledge is interpreted as connected information and information becomes knowledge if it is related to a certain context and is useful for it. Some representatives even define knowledge as information which is relevant and is new from the perspective of the information receiver (Rechenberg, 2006). Within the interpretation of having an *information sender*, an *information channel*, and an *information receiver*, *information* can be considered a *message* being sent from the information sender to the information receiver via an information channel. Accordingly, often the terms “data” and “information” are used synonymously in information technology domains (Kuhlen, Seeger, and Strauch, 2004; Krcmar, 2015).

From the AI perspective, the particular focus of definitions lies on the efficient provision of different forms of knowledge in IS. This goes beyond knowledge as information. As AI systems are constructed, the key issues are the development of knowledge-based systems as well as effective organization and linking of information. Here, one can find three kinds of approaches. First, approaches of the *orthodox AI*, which symbolically map knowledge about the real world to a computer with the help of data engineers. Second, there are approaches of the *non-symbolic AI* which are oriented to the computer-based simulation of neuro-scientific concepts and do consider knowledge implicitly. Third is attempts to combine the first two kinds of AI (cf. the concept called *symbolic distributed processing* by Peschl, 1990, p. 112).

Learning: Contrary to philosophical knowledge conceptions, AI-based approaches deal with approximate *values* than with an *objective truth* so that they aid to control processes computationally (Colburn and Shute, 2010). Some definitions explicitly assume statistical models to deal with uncertainties (Ritter, Martinetz, and Schulten, 1991, p. 265). A learning therefore refers to the optimization of these values with respect to an optimization objective so that uncertainties are reduced and the performance of AI-based systems is improved. Finally, this is similar to the knowledge implementation required by the KM domain. While a knowledge can be successfully measured and it becomes clear as to what extent far an organizational objective was met, in AI-contexts, this is motivated and enabled by the computational and

algorithmic processing; after the learning has been finalized, it becomes clear as to what extent an AI objective was met.

2.2.1.1.7 Neuro-Scientific and Cognitive-Scientific Definitions

Neuro-scientific approaches define knowledge as information, which is stored, integrated, and organized inside the brain (Solso, 2005, p. 242). Here, the same understanding about information can be found as applied by pedagogical and cognitive-psychological domains. Therefore, the human brain functions as a knowledge base which carries structured information. Contemporary approaches still struggle to interpret this structure, but for the system itself, the organization is plausible. When selecting, comparing, evaluating, and drawing consequences, knowledge is created from the resource of information (Reinmann-Rothmeier and Mandl, 2000).

Following cognitive concepts of a *radical constructivism*, a perception is related to the construction of mental *invariants*, which allow an organism to assimilate experiences (Peschl, 1990, p. 115). Accordingly, learning refers to better or more effective invariants of perception so that a behavior can be created which improves conditions for the survival of the learning organism.

As computers are used to simulate neuro-scientific and cognitive concepts, commonly, computational models are intended to model the functioning of real brains (Ritter, Martinetz, and Schulten, 1991) and thus intend to store knowledge as well. Peschl states that these models have *static knowledge* in the form of weights and *dynamic knowledge* in the form of neuronal activation (Peschl, 1990, p. 53). While the first kind of knowledge is responsible for the generation of dynamic knowledge, the latter is generated as the ANN is activated. Since the knowledge is distributed over the entire neuronal system, ANNs are considered as non-symbolic systems with nontransparent knowledge distribution (Peschl, 1990, p. 135).

Grum and Gronau consider a trained ANN as a knowledge model from which the identification of knowledge is very complex but would lead to an increase in interpretability and enables new kinds of process optimizations (Gronau and Grum, 2017; Grum and Gronau, 2018a). They assume knowledge to be a neuronal pattern that evolves over a certain period of time and causes a specific behavior of consecutive neurons.

Learning: Independent from the specific neuro-scientific and cognitive-scientific approach to define knowledge, learning refers to the change of neuronal parameters modeling the brain (static knowledge) such that the interplay of the entire system results in improved behavior (dynamic knowledge) measured by its performance (Peschl, 1990; Grum and Gronau, 2018a).

2.2.1.1.8 Critical Appraisal of Knowledge Definitions

Considering a lax definition of knowledge that does not even differentiate between knowledge and information, everyday knowledge conceptions are not suitable for the construction of a CoNM because the latter requires possibly formalized definitions for formalization and algorithmic processing. Here, the possibility to connect everyday understanding with the view on ANN is essential because ANN-based systems are intended to be considered as knowledge carriers. Since the CoNM shall consider both human as well as machine-based knowledge carriers, a broad understanding can be assured.

Building on computational models in the first place, IS-based knowledge conceptions are preferable. Since the real brain is intended to stand as a model for the construction of a CoNM, neuro-scientific mechanisms and cognitive concepts seem appropriate as technical foundation for the CoNM. Thus, the corresponding knowledge definitions must be considered as well. Fortunately, these definitions are all built on the processing of information, while the first deals with data streams of IS having a certain context; the latter deals with sensory information following biological principles.

As the CoNM intends to integrate neuronal processes within organizational contexts, knowledge conceptions from the domain of KM and KL needs to be considered further. More precisely, this refers to the integration of the aspects of *contextual meaning*, *dynamic nature*, *tacit dimension*, and *social construction*. Since these do not operate on the neuronal level, their integration is not trivial. Considering the intention to consider neuronal mechanisms on any process modeling abstraction level, the distinction of various knowledge definition levels is not an option. The option to use KM and KL-based knowledge conceptions as the only foundation is not adequate either because their abstract definition level of knowledge referring to fuzzy skills, cognition, and capabilities does not consider knowledge granularities as they are required on a very detailed neuronal level.

From a philosophical perspective, knowledge conceptions referring to *opinions* need to be very flexible to justify a neuron having an option. Considering neuronal systems consisting of various sub-systems, which can approximate human behaviors on behalf of one large, complex ANN, the idea of ANNs justifying their opinion becomes plausible. Therefore, each neuron takes part in the opinion-building process, which does not necessarily require an active role.

Having a *realistic* philosophical position, ANNs either obtain knowledge, which is referred to as internalism, or do not obtain knowledge at all because of being unable to justify their opinion about a certain issue. The latter is referred to as externalism. The externalistic argument can be weakened by the following: Although ANNs are considered non-symbolic systems having a nontransparent knowledge

distribution, contemporary approaches increase their interpretability because, for instance, of feature visualization (introduced in section 2.5.7.2). This increases the possibility to justify the opinion of ANN through anyone or any system.

As is commonly accepted in philosophy topics, the search for an adequate opinion-building process is a challenge. Although neuronal mechanisms are strictly formalized and would create the ANN, leading to *static* and *dynamic* knowledge as was described in (section 2.2.1.1.7), any ANN must stand the limitation of being only valid in its currently trained *context*. In this case, such as in philosophy, *context-based* knowledge conceptions are attractive. Similar to the *generality problem* issued in philosophical contexts (described in section 2.2.1.1.2), the endeavor for higher validity levels can be identified in neuronal contexts.

From an educational perspective, competence-based knowledge definitions are valuable because of the limited possibility to identify a robust opinion-building process. Similar to the characterization of valid contexts, the use of competences supports the characterization of ANNs for them to be successful in a specific situation.

From a pedagogical perspective, definitions are suitable to be considered because of the adequate use of ANN with respect to its capacities. Further, it is considered if the knowledge to be considered is a complex data set restricted by specific burdens. So far, pedagogical issues have not been considered by ANN concepts. This marks a potential use of a human or non-human pedagogue particularly adjusting the amount of information to be learned with respect to the biological circumstances of the ANN during the first learning phase. Thus, neuronal schemata can be built efficiently, which includes dealing with capacity issues (cognitive load theory following Sweller, Merriënboer, and Paas, 1998; Clark et al., 2006). If present one day, during the second learning phase, emotional factors of individual ANN learners (Lufi, Okasha, and Cohen, 2004) can be considered separate from the sustainable anchoring of schemata in the *long-term memory*.

2.2.1.9 Interim Conclusion

Building on Grum and Gronau's processual understanding of knowledge (Gronau and Grum, 2017; Grum and Gronau, 2018a), considering a context-based, philosophical conception and integrating it with KM-based definitions, this research defines knowledge as follows:

Definition 13 (Knowledge).

Knowledge is referred to as the time-based pattern of at least one knowledge carrier, known as schemata or invariant, that is evolved as soon as an activation is processed

when either data or information is perceived or requested by higher order mental activities. It is capable of affecting the realization of an actual entity within a subjective or objective reality, and this realization is made by the corresponding knowledge carrier(s).

The term “at least one knowledge carrier” in the definition presented considers the social dimension of knowledge to be constructed in a network of several, which refers to the *social construction* aspect from the KM domain. Thus, knowledge can be constructed on an individual level or an arbitrary complex system of knowledge carriers. Implicitly, this requires a hierarchy of patterns. The more basic the patterns of individual knowledge carriers considered, the more complex and higher-order patterns of knowledge carrier systems constructed. This is in agreement with the educational definitions of Bloom et al., who mention five levels beyond the knowledge level (Bloom et al., 1956). Each higher level refers to more complex patterns demanding more complex and laborious mental processes.

As the mentioned definitions do not specify the meaning of the term “knowledge carrier”, it is appropriate to recognize various aspects as knowledge carriers. The following presents only some examples: First is any ontological level required from the KM domain such as individuals, teams, departments, organizations, and so on. Second is any philosophical subject following an opinion-building process. Third is any biological system modeled by ANN mechanisms, such as a neuron or system of neurons, among others.

Since the definition mentions a *time-based pattern*, the *dynamic nature* required by the KM domain is considered. As both single moments as well as a period of time, which refers to a collection of ordered moments, are addressed by the term “time-based pattern” as well, the *dynamic knowledge* of ANN is considered. Thus, the definition allows the consideration of a period of time of an undefined length in order to make a pattern recognized as knowledge. This might be denominated by an interpretable term so that it can be recognized by humans, but it does not necessarily have to. The question is whether an easily understandable denomination can be identified by humans for any pattern at all. Ideally, this is in harmony with an everyday understanding of knowledge.

Besides the dynamic knowledge mentioned by ANN definitions, the ANN-typical knowledge form recognized as static knowledge is not addressed explicitly by the definition. Although knowledge can be extracted from the weight design of an ANN, the structural elements of an ANN are not interpreted to be knowledge by the definition presented. Its structure will be identified as knowledge in exactly that moment when a pattern is constructed during the act of the analysis of structural elements. Therefore, the structure is interpreted as an environmental condition only.

With a focus on a neuronal level, the *tacit dimension* is considered because of the following reasons: if knowledge cannot be externalized on an organizational level presented by the KM domain; if it is intended to be identified on more granular neuronal processes. On the neuronal level itself, any knowledge which refers to the distributed and non-symbolic knowledge is considered tacit.

When the definition mentions the “realization of an actual entity”, the situational effect of knowledge is addressed, for e.g., on the realization of an activity, task, and so on. In this case, the requirement of the problem-oriented situational implementation from the KM domain is satisfied, which demands knowledge used in the form of a solution. Further, with this, the requirement to be testable from educational knowledge definitions is fulfilled.

Because of the utilization of the term “actual entity” in the definition presented, the following two associations are considered: first, the relation to the definition of process models (Def. 3), and second, the understanding of knowledge as a *processual phenomenon* from the KM domain.

Philosophical knowledge conceptions have not been considered entirely because they are mutually exclusive. Hence, the definition presented aligns only with a selection of most relevant philosophical schools, which is described in the following. For this purpose, it is interpreted that neurons and systems of neurons will produce an *opinion* if they have been trained. The act of training refers to a robust opinion-building process which is valid for exactly the *context* trained. Attempts for generalization therefore broaden the context trained. An opinion-building process further can refer to the following three: first, educational attempts to build an opinion on behalf of competences; second, management attempts to build an opinion regarding organizational objectives; third, pedagogical attempts to build an opinion regarding learning problems, individual learning capacities, as well as learning burdens.

Here, an externalistic perspective was selected because contemporary ANNs are not capable of justifying their knowledge by themselves. The CoNM therefore intends to identify knowledge that has not been identified by humans before. This, for instance, makes the CoNM suitable to rediscover simple natural laws, such as physical laws of motion, gravity, or mechanics. Further, this makes the CoNM ideal to discover complex natural laws, such as string theories or quantum mechanics. Thus, this would hopefully allow researchers to identify solutions they have not considered before (see *integration constant attributes* of Fig. 2.1).

The term “subjective or objective reality” associates the definition of knowledge with the definition of model (Def. 1) and the act of modeling (Def. 5). Since the CoNM intends to create models of knowledge, it must differentiate between knowledge representations of an original system environment and its model system environment. The more or less subjective perception and acting of the model con-

structor during the act of modeling of both kinds of environments is considered in this case so that the definition serves in both kinds of environments as a valuable definition of knowledge.

The terms “schemata” and “invariant” mentioned in the presented definition refer to commonly known denominations of pedagogical and neuro-scientific contexts. As a common denominator, they bring together definitions of different contexts. Further, the term “patterns” brings various domains together. As patterns can refer to fuzzy abstractions, definitions of an everyday understanding of knowledge as well as manageable knowledge abstractions can be considered, which are required by the KM and KL domains. Moreover, as patterns can also be built on values, the definition is in harmony with IS-based knowledge conceptions.

Although Weinert recommends knowledge use along with the related motivational, volitional, and social willingness and abilities, these kinds of topics have not been considered by the definition and the first version of a CoNM as they require higher-order mental processes. The first CoNM version therefore focuses only on the technical functioning.

2.2.1.2 Knowledge Representation Formalisms

Considering different forms of knowledge, an essential question in computer science (CS) and knowledge management (KM) concerns the digital representation of knowledge in computer systems. Particularly, the CS domain of AI intends to store and develop knowledge through computer systems, and a literature search of the term “knowledge representation” identified a collection of different formalisms to represent knowledge. The following provides an overview and appraises attempts critically.

Logic-based Knowledge Representation: Logic-based representation mechanisms allow knowledge to be represented by two kinds of logic: first, propositional logic (zeroth-order logic) and second, predicate logic (first-order logic). Both kinds refer to formal languages being able to represent and reason with knowledge (Lämmel and Cleve, 2012, p. 69).

The first establishes knowledge as *atomic propositions*, which are either true or false, can be connected by *logical connectives* and thus form more complex propositions (Lämmel and Cleve, 2012, p. 33). The second establishes knowledge as *atomic formal systems* of non-logical objects, predicates about them, and quantifiers. These can include atomic propositions of the zeroth-order logic. Further, they can be combined with logical connectives and quantifiers to form more complex systems (Lämmel and Cleve, 2012, p. 52).

As zeroth-order logic is considered by formal PMLs and first-order logic entails numerical systems, the ANN can be interpreted to be numerical systems and logic-based knowledge representation mechanisms are suitable as CoNM foundation.

Rule-based Knowledge Representation: Knowledge representation mechanisms going beyond logic-based knowledge representations and allowing the use of non-logic relations as they are used in everyday life, referred to as rule-based representation mechanisms (Lämmel and Cleve, 2012, p. 71). Here, *rules* refer to *inferences* and *actions* that are related to the fulfillment of certain *conditions* and follow a common structure:

$$\text{IF condition THEN inference.} \quad (2.7)$$

In business contexts, these rules refer to *business rules* and explicitly present organizational knowledge about processes and conditions (Lämmel and Cleve, 2012, p. 82). Further, *facts* refer to the current state of a system. If rules and facts are linked, *requests* representing a problem can be processed such that a problem solution is derived with the aid of rule-based and fact-based knowledge.

Although the preparation of knowledge is strenuous and the selection of rules must be guided in bigger problems, the following beneficial characteristics are present: First, as knowledge is represented explicitly, it enables an automated processing of knowledge, which is preferable in business contexts as management routines can be established. Second, rules are comprehensible and inferences are explainable. Third, rules can be combined so that knowledge is constituted in a modular manner.

As PMLs consider everyday knowledge as well, rules are preferable to represent processual relations and facts to represent current process instances that are issued by the CoNM. Rules that can be combined make rule-based attempts to represent knowledge especially ideal for a CoNM foundation, since models can be considered on different levels of abstraction and granularity. The CoNM can thus be harmonized with different levels of rule combinations.

Fuzzy Knowledge: Distinct from bivalent logical statements requiring a *truth value* of either *true* or *false*, knowledge tends to be vague and uncertain (Lämmel and Cleve, 2012, p. 91). Therefore, knowledge can refer to a *partial truth* providing values between the range of completely true and completely false (Novák, Perfilieva, and Mockor, 1999).

Inspired by the probability theory, partial truth values represent objective probabilities about the certainty of an event that occurs or a statement being true or subjective degrees of someone's beliefs about it. Hence, we have

$$CF(H \mid E). \quad (2.8)$$

Here, CF refers to the Certainty Factor, which represents the trustworthiness that the statement H is true given a certain observation or belief E .

According to the fuzzy set theory, partial truth values can further represent an attribute belonging to a certain set which follows a *membership function* (Lämmel and Cleve, 2012, p. 102). With this, the management of subjective and overlapping linguistic denominations is enabled.

Although dealing with uncertainties is a fruitful complement for the final result, a first CoNM version will not present uncertainties: its focus shall lie on the mapping of models constructed by the final product and its representation within the original environment. Furthermore, observations either belong to an atomic process activity or not; this does not determine the relevance or weight of that dependency. The fuzziness will not be considered in a first CoNM version either. Hence, the association of models constructed by the final product and its representation within the original environment are assumed to be clear and non-overlapping.

Knowledge Maps: Considering the different types of knowledge maps, they all provide structured and graphical ways to visualize the knowledge of an organization. Further, they provide references to the corresponding knowledge source in reality (Juengst and Strittmatter, 1995; Eppler, 1997; Mandel and Fischer, 2000; Eppler, 2004; Tergan, 2004; Wiig, 2004; Young and Organization, 2010). According to Eppler, there are five different types of knowledge maps (Eppler, 1997). First, *knowledge source maps* present experts with the aid of self-chosen criteria. Second, *knowledge asset maps* visualize the knowledge portfolio of different ontological levels of an organization. Third, *knowledge structure maps* present the structure of a knowledge domain by logically connected areas and their relations. Fourth, *knowledge application maps* relate knowledge to certain process steps and business events. Fifth, *knowledge development maps* visualize steps required to access a certain knowledge level.

Although each type of knowledge map would establish an attractive complement of the CoNM, the maps are not required for the key technique involved in the final artifact. Hence, they are not considered as its foundation yet.

Taxonomies: Taxonomies refer to representations of the formal structure of classes or types of objects within a domain (Hurwitz, Kaufman, and Bowles, 2015, p. 80). Since classification rules of a taxonomy are applied to all objects and instances of a system, they lead to complete, consistent, and unambiguous models. Hence, they constitute a hierarchical manner of codifying information and capture knowledge

within a particular field of investigation. Thus, they support the use of knowledge (Gronau, 2009, p. 92). Each taxonomic level represents a certain category and consists of subsets, with each inheriting all the properties defined in the superset.

Since taxonomies are preferable for the establishment of a joint reference framework for concepts relevant for the CoNM, they are attractive for its methodological construction. Since they can deal with hierarchies, and the final artifact intends to deal with process hierarchies as well, they are further ideal to visualize knowledge, which is codified in ANN structures and distributed over various process levels of abstraction and granularity.

Ontologies, Topic Maps, and Frames: Ontologies refer to representations of the formal structure of classes or types of objects of at least one domain (Hurwitz, Kaufman, and Bowles, 2015, p. 81). So, they establish a formal nomenclature and definition of the categories of several domains. This integrates properties and relations between the concepts, data, information, and knowledge so that the complexity of the possible term interpretations can be reduced (Gruber, 1993; Feilmayr and Wös, 2016; Gronau, 2009, p. 92).

Unlike taxonomies, ontologies do not require a strict hierarchical structuring, so that networks instead of pyramidal structures are built whose relations can have different implications (Lehner, 2014, p. 210). As standardized versions of ontologies, one can find *topic maps* (ISO Central Secretary, 2003) or *knowledge networks* (Lämmel and Cleve, 2012, p. 89). As uniform schemata are required for the storage of attributes, the concept of frames provides a standardized container to store knowledge, such as actual values, default values, and restrictions (Lämmel and Cleve, 2012, p. 88).

The CoNM intends to deal with knowledge representations coming from various contexts. According to the initial definition, the dealing within its field of action refers to one common understanding. Thus, ontologies are suitable if contexts are differentiated in later artifact iterations. As frames require fixed structures, they are suitable for a technical foundation, which excludes the clear visualization by underlying modeling languages.

Process Modeling: Having described process modeling adequately in section 2.1 and section 2.2, the model itself represents knowledge about the original environment and enables designable knowledge application in organizations (Lehner, 2014, p. 212).

As the CoNM builds on a processual interpretation of the reality (Def. 3), which is accompanied by processual knowledge understanding (Def. 13), process modeling techniques are highly preferable to stand as a foundation for the final artifact.

Simple Trees and Semantic Web: Logical data structures capturing parent-child relationships, referred to as simple trees, can be implemented as table and efficiently represent knowledge (Hurwitz, Kaufman, and Bowles, 2015, p. 83). Further, semantic network attempts try to evolve the current world wide web by adding semantic attributes so that everything on the web follows uniformity and is machine usable (Hurwitz, Kaufman, and Bowles, 2015, p. 84). Thus, knowledge is represented by semantics and is accessible by semantic approaches (Lämmel and Cleve, 2012, p. 89).

Simple trees and semantic web approaches are not attractive as foundation because of the following reasons. First, the CoNM does not only intend to reference knowledge and to make it accessible by attribute mappings. It rather intends to technically deal with knowledge, which goes beyond the simple use of semantic web structures and, for example, the use of simple tree structures by Internet users. Second, the final artifact intends to develop knowledge. Simple trees and semantic web approaches do not enable mechanisms such as biological processes or simulations; they refer to data structures only.

Interim Conclusion: Considering various approaches to represent knowledge, particularly the anchoring of knowledge in process modeling is attractive as foundation. A selection of corresponding representatives is considered in section 2.2.3. As this operates on a non-technical level, further, logic-based and rule-based mechanisms are suitable to technically support the construction of the final artifact. Here, attractive interpretations must be established considering with numerical values of ANN structures and attractive attempts of fuzzy logics.

2.2.1.3 Knowledge Transfers

Considering various forms and formalisms to represent knowledge (section 2.2.1.2), only some enable the dynamic characterization of knowledge, which refers to knowledge generation and its evolution over a period of time. Since even less representations enable the time-dependent anchoring of knowledge within a given process and no approach considers all the dynamic aspects of knowledge discussed before, the following section defines the concept of knowledge transfer processes, which enables the dynamic characterization and observation of knowledge on the basis of processes as the best tool available so far. Unfortunately, knowledge patterns according to Def 13 are not considered yet so that the concept of knowledge transfers will stand as the foundation for the CoNM construction. Since conceptions of knowledge transfers provide definitions focused on different aspects, the following presents relevant aspects for establishing a synthesized definition.

Knowledge Transfer Aspects: A definition of knowledge transfer has to consider the *transfer process* itself as well as its *content* (Gronau and Grum, 2019). The generation of knowledge serves as a starting point for knowledge transfer processes. An individual generates new knowledge with the help of perceptions, facts, events, feelings, and moral concepts, all of which influence the person permanently (Davenport and Prusak, 2000). In this context, knowledge is available in the person's environment and is transferred to the person. When individuals want to transfer their knowledge, parts of their mental models have to be made tangible for others. As knowledge is transferred between individuals, groups, departments, branches, among other things., an internal knowledge transfer takes place (Eisenhardt and Santos, 2002; Kriwet, 1997). However, the concept of knowledge transfers is not limited to internal knowledge transfers and can go beyond a system or organization's border as well (Ullrich et al., 2017). For the transfer of knowledge, various explication methods are used, such as speech, texts, pictures, graphics, and metaphors (Lutz, 2008, p. 97). Hence, from a knowledge management perspective, the transfer of knowledge is considered on various ontological levels, with persons considered as knowledge carriers on the most detailed level. Knowledge on a neuronal level has not been considered yet.

Codification Aspects: The codification of knowledge, and with this the simplifying conversion of knowledge to information, is part of the process of knowledge transfer (Liebowitz, 1999; Maier, Hädrich, and Peinl, 2005; Maier, 2007; Alexander and Childe, 2012, p. 5). The medium is not characterized in any way. Hence, a transformation of knowledge into written information, for e.g., and the subsequent reverse codification by the knowledge receiver through reading may be considered as a knowledge transfer equivalent to a knowledge transfer using spoken words alone. This interpretation allows for a concept of knowledge transfer that provides a fundamental basis for the consideration of complex, organization-wide, knowledge-intensive business processes (Gronau and Weber, 2004b). Further, this does not rule out knowledge transfers at a neuronal level, which are required for a CoNM integrating neuronal and organizational processes.

Conceptual Models: The content of the knowledge to be transferred in knowledge transfer processes can be differentiated by type. The relevance of the transferred knowledge can serve as a means of further differentiation (Justus, 1999, p. 157), as well as the extent of the knowledge and other content-based criteria (Wirth, 2012, p. 32). Various conceptual models can be found to describe knowledge transfers on the basis of differentiations (e.g. Shannon and Weaver, 1963; Nonaka and Takeuchi, 1995; Von Krogh et al., 2000; Disterer, 2000; Cummings and Teng, 2003; Minbaeva

et al., 2003; Peinl, 2006; Maier, 2007; Inkpen, 2008; Gronau and Grum, 2019). Since only Gronau and Grum (2019) discuss the velocity of knowledge transfers, and provide quantitative models which a CoNM must consider, corresponding models are considered in the following.

Quantitative Knowledge Transfer Operationalizations: Building on Minbaeva et al. (2003, p. 587), Gronau and Grum define knowledge transfer as knowledge-intensive processes referring to the identification of knowledge, its transfer from knowledge carrier to knowledge receiver, and its application by the knowledge receiver (Gronau and Grum, 2019). For the operationalization of knowledge transfers, the concept called *knowledge spiral* developed by Nonaka and Takeuchi (1995) can be considered.

Nonaka and Takeuchi (1995) consider the kind of ontological level and epistemological level in question to differentiate the kind of knowledge transfer. Fig. 2.19 visualizes this as follows.

As knowledge is transferred among ontological dimensions, for e.g., from individual to individual or from individual to a group, a *socialization* is realized. If knowledge is made tangible because of an *externalization*, it changes its epistemological dimension. Therefore, the person-bound *tacit knowledge* becomes the person-unattached *explicit knowledge*, which can be easily distributed within the system modeled. Accordingly, different objects of explicit knowledge can be *combined* to create new explicit knowledge. If a person deals with explicit knowledge, they integrate the explicit knowledge with its mental models, thereby creating new

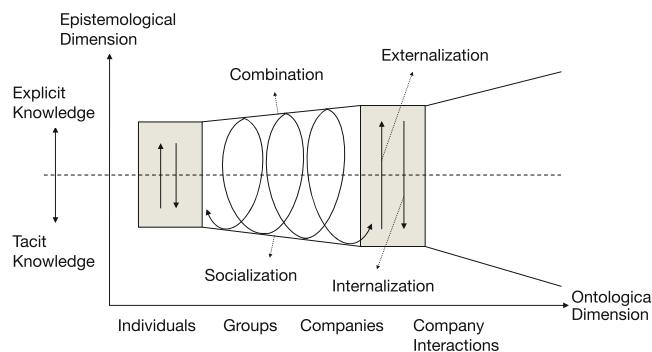


Figure 2.19 Knowledge Spiral (following Nonaka and Takeuchi, 1995)

tacit knowledge, which is called *internalization*. Although this model operationalizes knowledge transfers by their epistemological dimension, it does not consider knowledge generation from data and material objects, for e.g., on behalf of *analyses* and *interpretations* at all.

For the first four kinds of knowledge transfers, Gronau and Grum (2019) conducted empirical research with experiments. Since this kind of operationalization has been statistically verified, it is suitable as a foundation for the construction of a CoNM.

Critical Appraisal: Considering the knowledge definition stated before (Def. 13), knowledge is considered as a processual phenomenon itself, referred to as the *time-based pattern* here. Building on Gronau and Grum's processual understanding of knowledge transfers (Gronau and Grum, 2019) and its mapping to a model (Def. 1) constructed during the act of modeling (Def. 5), this research defines models of knowledge transfers or what is called knowledge process models as follows:

Definition 14 (Knowledge Process Model).

A knowledge process model (synonym for knowledge transfer model) is a process model of at least one process or a hierarchy of processes in which each process refers to actual overlapping, independent entities providing a temporal coherence and unity about the identification of knowledge, its transfer from knowledge carrier to knowledge receiver, and its application by the knowledge receiver. It can be measured by the right tools and is logical from the perspective of an individual as it is understood. It forms all together the actual existing world in a multiplicity and become real in the iterative interplay of concrescence in the private and satisfaction in the public.

As the definition does not specify the terms “knowledge carrier” and “knowledge receiver”, both can stand for the same or different human-based process participant(s) (coming from the process context), machine-based business process participant(s) (coming from the business process context), neuronal representative(s) (coming from the ANN context), or any other entity of knowledge-related *actual entity*.

Contrary to the time-dependent definition of knowledge (Def. 13), the *knowledge process model* enables the processual description of *time-based patterns*, *schemata*, and *invariants*, which does not refer to the pattern itself. It rather unfolds complex patterns through more simple patterns so that a sequential effect becomes apparent. Accordingly, the definition provides a possibility to describe any representation of knowledge (section 2.2.1.2) with knowledge transfer processes on more detailed

levels with atomic patterns. Therefore, different kinds of knowledge-building blocks are constructed which can be combined to construct more sophisticated components. Further, it allows the embedding of knowledge in processes and business processes, because of the term “actual entity” as a common denominator. Owing to this, it is in harmony with the definition of process models (Def. 3) as well as business process models (Def. 4) and allows the modeling in the sense of Def. 5 on base of any process modeling language (Def. 8).

Since both *knowledge* and *knowledge processes* are recognized as models in the sense of Def. 1, the constructive and formalized operating with these two abstract and intangible entities is enabled. This includes the description, simulation, analysis, design, and evaluation on the basis of more or less subjective interpretations, limited perceptions of an object system by and purpose-related design decisions of the model creator *mc*.

The statements “logically from the perspective of an individual” and “as it is understood” of the definition presented present the challenge to denominate knowledge and to logically relate the time-dependent use within a knowledge transfer model. While the knowledge process model makes sense for the one, it is not plausible or logical for the other, which is exactly the point at which interpretation starts.

Because of the recognition of *knowledge* and *knowledge processes* as process models (Def. 3) as well as business process models (Def. 4), the purpose-related creation of viewpoints of one or more *stakeholders* or ANN-based systems (denominated here as *ANN instances*) in the form of perspectives (Def. 6) is enabled. This further allows the independent choice of knowledge representation formalism and the transformation among different formalisms (section 2.2.1.2).

Since *knowledge* and *knowledge transfers* are recognized as processes, they can be interrelated with enterprise architectures (EA) according to Def. 7. This means the following. First, they are embedded in data, hardware, software, and communication resources, which means that the knowledge process model is fed with EA data, is carried out on EA hardware and software, and considers EA process participants and available communication structures. Second, they are allowed to refer to EA activities, which refer to the organization of data, hardware, software, and communication resources. Here, the organization of organizational entities following biologically plausible mechanisms of the brain is enabled.

Although, the definition does not exclude knowledge transfers across company boundaries (required by Holden and Von Kortzfleisch, 2004, p. 130, and Lam, 1997), these are not considered in this research since this kind of transfer also entails on-building questions of the security of intellectual capital (Chesbrough, Vanhaverbeke, and West, 2006; Lin, 2007).

Overall, on the basis of the definition of knowledge transfer presented, knowledge can be considered as dynamic phenomenon and visualized over time. Here, the CoNM is intended to consider processual circumstances as well as processual knowledge generation, manipulation, and utilization. Since this consideration might be visualized on behalf of knowledge modeling languages (introduced in section 2.2.3), a knowledge modeling and process modeling can be considered jointly, which is the foundation of efficient knowledge management (introduced in section 2.2.2). Here, if neuronal mechanisms are used in order to create process models, the human-made management of knowledge can be observed in analogy to the biological organization principles of the human brain. Therefore, the CoNM constructed will be the foundation for the transfer of knowledge management mechanisms to the biologic treatment and mechanisms from the latter to the first treatment.

2.2.2 Knowledge Management Concepts

In accordance with the knowledge understanding presented in section 2.2.1.5 (KM-based and KL-oriented knowledge understanding), organizational principles focusing on the efficient management of knowledge are referred to as *knowledge management* (KM). This is closely connected with operational questions about the identification of knowledge, its documentation and successful utilization, as well as questions regarding the strategic development of a company member's knowledge, its efficient organization, control regarding a sustainable knowledge use, and implementation of measures to improve knowledge-based issues (Gronau and Grum, 2019).

Selection of representatives: A literature search of the term “knowledge management” identified various concepts of *knowledge management* providing more or less sophisticated approaches adequate for a knowledge modeling. The following presents a selection of more than 160 concepts (Heisig, 2009) and clusters the main representatives. Although various attempts have been realized to systemize them (Begoña Lloria, 2008; Sheffield, 2009; Heisig, 2009) and rankings about KM model candidates have been prepared (Serenko and Bontis, 2009; Bontis and Serenko, 2009; Ergazakis, Metaxiotis, and Askounis, 2013), a converging of KM models and KM concepts has not yet occurred (Lehner, 2014). The following therefore shall serve as the preselection of prominent representatives per category.

- *Holistic KM models* refer to concepts that unite multiple KM relevant aspects. Examples can be found at the Know-Net-Framework, which involves ontological

levels and relevant infrastructure elements (Mentzas et al., 2001), and the cyclic KM (Probst, Raub, and Romhardt, 1997) providing relevant KM building blocks (Probst, Raub, and Romhardt, 2006).

- *Knowledge market models* issue the demand and supply of knowledge by market-similar approaches. Examples refer to North's model based on framework conditions and rules for the knowledge market (North, 1998). Further, models are also based on different kinds of incentives on these kinds of markets (Frischmuth, 2002; Desouza et al., 2005; Zhang and Sundaresan, 2010).
- *Constructivist KM models* focus on the creation of knowledge and the support of learning individuals. Examples can be found at the SECI-model incl. its knowledge spiral (Nonaka and Takeuchi, 1995), and approaches focusing on the results of mental construction processes (Meinsen, 2003).
- *Process-oriented KM models* focus on the dealing of knowledge right within the process. KM measures are therefore oriented according to the underlying processes models. First attempts can be found at the model called Improving Knowledge Work Processes (Davenport, Jarvenpaa, and Beers, 1996), that involves dealing with knowledge in processes but does not show concrete modeling attempts. Prominent examples of this kind of anchoring are based on Scheer's approach (Allweyer and Scheer, 1995; Allweyer, 1998), refer to PROMOTE (Hinkelmann, Karagiannis, and Telesko, 2002), GPO-WM (Heisig, 2002; Heisig, 2005) and the Potsdam KM Model (Gronau, 2009). Further examples consider reference models for the design of knowledge intensive processes (Warnecke, Gissler, and Stammwitz, 1998; Remus, 2002; Mertins, Kohl, and Orth, 2016) and focus on the support by adequate IT systems (Integrated Design of Knowledge Management Systems, Nissen, Kamel, and Sengupta, 2000). The model called Business Knowledge Management focuses on the value creation and knowledge organization in process models (Bach, Österle, and Vogler, 2000). Informal communication activities apart from business processes are described by the approach called Communication Diagnosis (Dämming, Hess, and Borgmann, 2002).
- *Problem-oriented KM models* focus on the easy and efficient dealing with knowledge and the overcoming of problems at this kind of dealing (Wildner, 2011).
- *Person-oriented KM models* focus on the management of individual knowledge carriers. Examples can be found at (Mandl and Reinmann-Rothmeier, 2000; Cheong and Tsui, 2011; McFarlane, 2011).
- *System-oriented KM models* focus on the target-oriented development of by sub-structures manifesting unities. Examples can be found at (Willke, 2007).
- *Social KM models* address the aspects of individuals in social networks (Wendt, 1998) or communities (Munich Knowledge Management Model, Reinmann-

- Rothmeier, 2001 and Enterprise Knowledge Media Reference Model, Eppler et al., 2000).
- *Collaborative KM models* have a focus on the use of new technologies for the purpose of KM (Kain, 2010; Garg, Smith, and Telang, 2011; Warta, 2011).
 - Technique-oriented KM models focus on technical realization by KM tools (Richter, 2008).
 - *IP-oriented models* assume knowledge to be managed as intellectual property (Schneider, 1996; Schnabel, 2013).
 - *Project-oriented KM models* focus on the knowledge transfer from project to project so that project team members profit from this kind of transfer (Olesnitz and Busch, 2008; Hanisch et al., 2009; Yanzer Cabral, Ribeiro, and Noll, 2014).
 - *Specialized KM models* establish a particular focus on small and medium-sized enterprises (Lee and Lan, 2011), branches (Pirkkalainen and Pawlowski, 2013) as well as topics such as the organizational learning (Dierkes et al., 2001; Lau and Tsui, 2009).

Following Remus, there are only two important kinds of knowledge management (Remus, 2002): The first focuses on the management of knowledge as resource so that knowledge demands to be administrated. The second focuses on the management of knowledge utilization so that knowledge is appreciated to have a value and be part of the entire value chain of a company (Gronau, 2009, p. 9). The latter is referred to as *process-oriented KM* (Scheer, 1998; Becker et al., 2000; Remus, 2002; Fettke and Loos, 2004b; Gronau and Müller, 2005; Gronau, Müller, and Korf, 2005; Hinkelmann, Thönssen, and Probst, 2005; Mertins, Kohl, and Orth, 2016).

Generally, any KM concept presented provides the following two CoNM potentials: first, a potential as a foundation for the CoNM and second, a potential for the inclusion of organizational concepts based on neuronal mechanisms. On behalf of the CoNM, the transfer of organization principles among both domains shall be enabled. As it was possible to identify on behalf of the CoNM how knowledge can be organized in companies following biologically plausible and over centuries evolutionary optimized principles, an economic potential can be assumed to exist. Further, as it was possible to identify on behalf of the CoNM how dynamic knowledge can be organized in neuronal structures following KM principles, it can be assumed that there is potential regarding the powerfulness of ANNs and their interpretability. The following examines prominent representatives regarding the extent to which biological principles of brains have been considered. The contribution therefore selects KM concepts according to the following criteria.

1. Considering knowledge conception established in section 2.2.1.1, only representatives considering the concept of *knowledge transfers* (introduced in section 2.2.1.3) have been considered. This puts a focus on process-oriented KM.
2. KM concepts having a broad acceptance in the KM community.
3. KM concepts being applied in at least one scientific publication.

Adequate examination level: In order to satisfy requirements for an adequate management of objects of investigation (OoI) in the terms of SLR (Levy and Ellis, 2006) which corresponds to the appliance of Bloom's taxonomy (Bloom et al., 1956), each KM concept is described by the following categories:

- The short *abstraction* serves as a kind of introduction to KM concepts on the same, abstract level. Therefore, it is presented at the very top of each sub-section.
- Its individual *description*, characterizes the individual OoI.
- *Side facts* describing the OoI-specific development and historic evolution so that the standing within the scientific community becomes clear.
- The provision of a *model* visualizes KM activities and its underlying context. This is completed by *explanations* clarifying how the model of the OoI is applied so that a concrete KM can be carried out.
- An *example* visualizes the appliance of the OoI and underlines management strengths of the OoI.
- Finally, the *critical appraisal* evaluates whether and how the individual KM concept is suited to stand as foundation for the CoNM.

Thus, according to Bloom's Taxonomy, categories considered for any KM concept correspond to the levels of *knowing* and *comprehending* if OoIs are described and side facts are presented. Further, when the level *applying* as an abstract KM model is applied and an example is presented, *synthesizing* as a common level of understanding is worked out as well as *analyzing* and *evaluating* as common analysis criteria are issued and the KM concepts are considered with respect to the CoNM.

2.2.2.1 Promote

PROMOTE is a management approach, which supports all the phases of a management cycle and includes strategic, planning, realization, and controlling tasks as relevant for adequate knowledge management (Hinkelmann, Karagiannis, and Telesko, 2002, p. 67). As a processual KM approach, it enables the realization of efficient knowledge utilization within the company's business process models and the creation of knowledge process models according to Def. 14.

Description: The Promote methodology provides five phases (Hinkelmann, Karagiannis, and Telesko, 2002, p. 74), which are oriented to the Business Process Management paradigm (planning, execution, controlling, and optimization) in accordance with Karagiannis (1995), Karagiannis, Junginger, and Strobl (1996). Each phase requires its individual KM activities. An overview of the phases and its activities can be found in the following.

1. *Aware Enterprise Knowledge*: KM objectives are determined. Further, core competencies, risks, and relevant business processes are identified.
2. *Discover Knowledge Processes*: Knowledge-intensive tasks, relevant knowledge carriers, knowledge flows, and the form of initial and resulting knowledge objects are modeled in current business process models (Spies and Trojan, 2007). Further, knowledge management instruments are determined.
3. *Modeling Knowledge Processes and Organizational Memory*: Knowledge flows identified are supported by the modeling of adequate, process-oriented knowledge models. These focus on the support of knowledge flows. Relevant knowledge is identified by meta-information about the processual context (Spies and Trojan, 2007), such that organizational knowledge can be provided efficiently.
4. *Making Knowledge Processes and Organizational Memory operational*: During the process execution, KM instruments are coordinated as a fixed component of daily routines. Hence, their operational realization is integrated with business process realizations (Spies and Trojan, 2007).
5. *Evaluate Enterprise Knowledge*: Changes of business processes and knowledge carriers are evaluated. Further, this involves a continuous optimization of processes (Karagiannis and Woitsch, 2002).

For adequate KM realization, Promote proposes twelve different kinds of model types (Karagiannis and Woitsch, 2002, p. 42): models issuing business process-related relations (business processes, working environments), models drawing attention to knowledge processing issues (skill documentation, knowledge structure, knowledge resource pools, knowledge process models, knowledge management processes, security models, workbench models), and models providing an overview (knowledge landscape, community model, process pool model).

With the help of these models, four different kinds of knowledge flows can be modeled (Hinkelmann, Karagiannis, and Telesko, 2002, p. 70): first, external knowledge transfers (e.g. consultants, trainings, Internet), second, knowledge transfers within the same process instance, third, knowledge transfers within different process instances, and fourth, knowledge transfers within different business processes. Although any process modeling language can be used to visualize these

transfers, Promote is by history linked with the modeling language (introduced in section 2.2.3.1) because of their joint origin. Further, it is accompanied by a process-based KM tool, which integrates various KM and modeling tools (Hinkelmann, Karagiannis, and Telesko, 2002, p. 68). Thus, a process modeling is enabled similar to the tool called *Adonis* (Junginger et al., 2000; Hinkelmann, Karagiannis, and Telesko, 2002).

Side facts: The Promote approach was developed in the EU project having the same name (Karagiannis and Telesko, 2000; O'Brien, 2010) by Prof. D. Karagiannis from the *BPMS Group* of the Institute of Informatics and Business Informatics at the University of Vienna (Drawehn et al., 2014) and the following industrial partners: The *BOC GmbH*, which is a software and consultancy company and the *FIDUCIA AG* and the *INTER-AMERICA* who have been pilot users. After its software prototype development, the *BOC Information Technologies Consulting AG* overtook the development and now, Promote software can be used in combination with the tool called *Adonis* (introduced as OoI in section 2.6.1.3).

Model: The components of the Promote KM model are summarized in Fig. 2.20 in accordance with Karagiannis et al. (Karagiannis, 1995; Karagiannis, Junginger, and Strobl, 1996; Karagiannis and Woitsch, 2002; Hinkelmann, Karagiannis, and Telesko, 2002; Spies and Trojan, 2007). By the provision of building blocks for adequate knowledge management, any measure can be characterized by the following two: *phases*, which have already been introduced before and are represented by ellipses, and different kinds of *knowledge management processes* as agenda items. The corresponding six kinds of knowledge management processes can be found at the center of the visualization and described thereafter.

Promote focuses on the management on behalf of six kinds of knowledge management processes described in accordance with Karagiannis and Woitsch (2002, p. 38).

1. *Knowledge model building processes*: This kind contains processes modeling and validation of knowledge models and includes their analysis.
2. *Knowledge identification processes*: This kind contains processes analyzing business processes with respect to knowledge-intensive activities and the examination of skills and competences required for these activities.
3. *Knowledge access processes*: This kind contains processes which focus on interactions of knowledge carriers and the following two: first, the organizational memory storing knowledge, and second, the Internet.

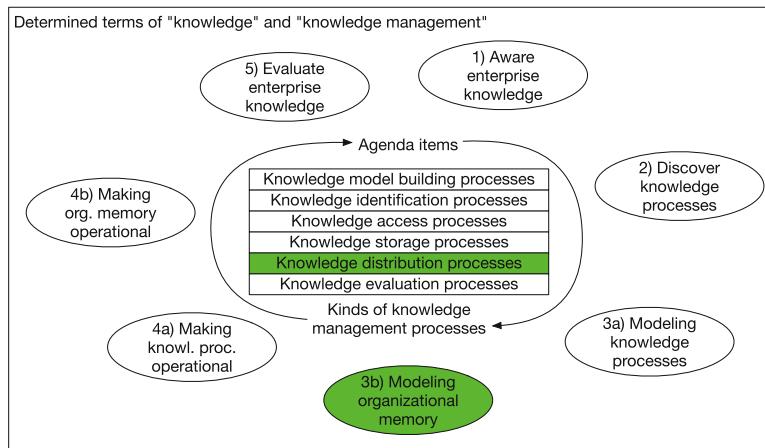


Figure 2.20 Promote Knowledge Management Model (highlighted elements issued in example)

4. *Knowledge storage processes*: This kind contains processes for the creation of textual knowledge annotations stored in the organizational memory and their categorization.
5. *Knowledge distribution processes*: This kind contains processes that support the generation, validation, and transfer of new entries in the organizational memory.
6. *Knowledge evaluation processes*: This kind contains processes defining evaluation criteria and monitoring of knowledge management processes according to the criteria defined.

Example: An example measure characterization of the Promote KM Model is described in the following. It exemplifies the dimensions and components highlighted in green in Fig. 2.20.

Example (Promote Knowledge Management Example).

A measure will be realized that refers to the identification, survey, and implementation of additional meta-information about knowledge items which are stored in the company's organizational memory and allow the optimal provision of knowledge to knowledge workers. Their current knowledge requirement during the execution of a certain product development process step is determined on behalf of the ERP

software: knowledge workers are asked to document their current process step so that relevant knowledge can be displayed beside common ERP form templates.

The ERP software detects the location and process step of a process model at which a knowledge worker currently is active, and the knowledge process model operationalizes relevant knowledge flows among internal and external process models and process instances. Therefore, knowledge items stored in the organizational memory can be provided automatically by the ERP system. Since this entails the distribution of knowledge stored in the organizational memory, this constructs a *knowledge distribution process*.

As the design of the current organizational memory is enhanced by additional meta-information, this concerns activities of the phase called *modeling organizational memory*. The more meta-information of knowledge items available and the more relevant the selected meta-information, the more helpful and effective is the knowledge distribution realized.

Critical appraisal: Since Promote builds on current process models of a company's business processes, the following three are addressed implicitly by the underlying process modeling language: organizational elements, the processual behavior, and functional elements. Requirements about the explicit modeling of relevant modeling entities are not specified any further. Hence, modeling language criteria from the business process domain cannot be satisfied.

Modeling language criteria from the KM domain are addressed by convention as Promote demands the discovery and operationalization of KM processes. Although the specific forms and types of knowledge are not specified any further, this also depends on the choice of the underlying knowledge process modeling language, and the management of knowledge flows is enabled. Considering an organizational memory by conception, the use of information and knowledge can be addressed by the Promote KM Model.

Due to its original focus, neither the simulation of knowledge processes or business processes nor the use of ANN models are considered by Promote such that the corresponding modeling language criteria are not addressed at all. Moreover, relevant simulation and ANN mechanisms are not issued for use in tools. The use of tools only focuses on criteria for the modeling of knowledge processes, their operationalization, the implementation of an organizational memory, and knowledge evaluation. Here, corresponding tool criteria can only be satisfied on an abstract conceptual level as concrete mechanisms are not provided by Promote.

The Promote KM Model enables practical access for an organizational KM. Certain theoretically founded KM activities and kinds of KM processes make it

suitable as an interface for the CoNM on an organizational operation level. Unfortunately, the model's components can only serve as building blocks for a neuronal task realization because Promote does not provide a technical foundation. Hence, the use of Promote as the KM foundation for the CoNM requires combination with concepts providing the required technical foundation that enable the management of simulations and neuronal mechanisms.

2.2.2.2 GPO-WM

The GPO-WM is a management approach, which supports all the phases of a management cycle and includes strategic planning, realization, and control of tasks relevant for adequate knowledge management (Gronau, 2009, p. 66). As a processual KM approach, it enables the realization of efficient knowledge utilization within the company's business process models and the creation of knowledge process models according to Def. 14.

Description: The GPO-WM methodology provides eight phases according to Heisig (2002, p. 59), which enable the implementation of knowledge management, the evaluation of the as-is state of a company with the aid of an audit instrument (Mertens, 2001), as well as the derivation and realization of measures according to design conventions and best practices (Mertens, 2001, p. 59). An overview of the phases can be found in the following enumeration.

1. *KM Strategy Definition & Project Objectives:* The strategic relevance of knowledge management is determined and operationalized by the definition of concrete project objectives. Their fulfillment will be evaluated during the course of KM realization.
2. *Business Process Selection:* Knowledge-intensive business processes are selected for improvement with KM methods and instruments.
3. *Assessment of KM Conditions:* With the aid of Fraunhofer's audit instrument, initial parameters and information about KM conditions are surveyed (Mertens, 2001).
4. *Analysis of Knowledge-Intensive Business Processes:* Selected business processes are analyzed with respect to the fulfillment of the core activities of a knowledge management. This includes the use of resources such as knowledge carriers and information systems.
5. *KM Building Blocks & Scenarios:* Analysis results and potentials for improvement are addressed by best practices, checklists, work aids, and instruments, such as Yellow Pages, Wikis, and so on. These are combined with bundles and referred to as solution scenarios.

6. *Measure & Implementation Planning*: The realization of solution scenarios is operationalized by concrete measures so that the implementation can be planned accordingly.
7. *Pilot Solution Implementation*: Before organization-wide implementations are performed, solution scenarios are tested in smaller pilot implementations.
8. *Use & Evaluation*: The measures are realized organization wide and considered as new standard. The use in everyday life is evaluated and compared with objectives initially specified.

Side facts: The GPO-WM approach was developed by Prof. Peter Heisig from the *Competence Center* of the Fraunhofer IPK in Berlin (Heisig, 2002). The modeling can be carried out with the aid of the knowledge process modeling language (introduced as a separate OoI in section 2.2.3.2) and the software tool called Mo²Go.

Model: The components of the GPO-WM are oriented to the Fraunhofer reference model of knowledge management and are summarized in Fig. 2.21.

The Fraunhofer reference model focuses on business processes which form the application area of knowledge. In this case, the *processual KM activities* of *knowledge generation, storage, distribution, and use* are intended to be a closed circle

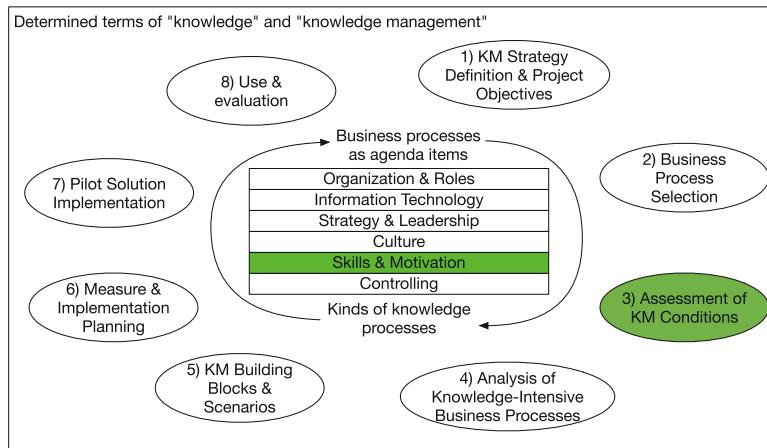


Figure 2.21 GPO-WM Knowledge Management Model (highlighted elements issued in example)

of activities (Heisig, 2002, p. 53; Heisig, 2006). Further, the activities *determinate knowledge objectives* and *knowledge identification* are part of this circle (Heisig, 2002, p. 53), which is visualized in the figure by the circular arrows.

Based on surveys, six *KM design fields* have been identified as relevant for the successful realization of knowledge management and therefore have to be addressed by its measures (Heisig, 1999; Heisig, 2000; Heisig, 2002, p. 54). These are visualized in the figure center and explained in the following:

1. *Organization & Roles*: Measures need to provide appropriate processes organizations and roles so that tasks are coordinated efficiently, provide the minimum number of interfaces, and the flow of knowledge is supported throughout the company (Mertens, 2001, p. 107).
2. *Information Technology*: Measures need to provide an appropriate infrastructure which supports communication among process participants, cooperation, coordination, and the access of information and knowledge. This includes, for e.g., data warehouse collection data, groupware, Internet, and Intranet allowing communication and collaboration, and knowledge databases and workflow engines enabling knowledge access (Mertens, 2001, p. 112).
3. *Strategy & Leadership*: Measures need to focus on an active management. This includes the establishment of an autonomous, self-determined action of process participants acting as coach, as well as the acknowledgment of employees (internal knowledge carrier) and external knowledge carriers (Mertens, 2001, p. 109).
4. *Culture*: Measures need to address the organizational climate so that a failure-tolerant, knowledge exchange-friendly, self-determined, and learning-friendly environment is supported (Mertens, 2001, p. 111). This includes the design of an adequate organizational structure.
5. *Skills & Motivation*: Measures need to enhance individual skills relevant for business process realizations. Further, they need to motivate employees to conduct effective KM activities. This, for instance, includes monetary rewards, expansion of the range of activities, and responsibilities, as well as the conducting seminars and trainings (Mertens, 2001, p. 108).
6. *Controlling*: Measures need to determine whether predefined targets have been realized successfully, which is mostly realized in terms of hard and soft indicators (Mertens, 2001, p. 113).

As the model's central elements, the eight phases are visualized by ellipses in the figure, which already has been described previously.

Example: An example measure characterization of the GPO-WM Model is described in the following with dimensions and components highlighted in green in Fig. 2.21.

Example (GPO-WM Knowledge Management Example).

A measure will be realized that refers to auditing on behalf of the Fraunhofer audit instrument in one-day workshops. Here, each research department will be surveyed about its proper internal knowledge-intensive research process. Since the workshop day binds personnel resources for one day, every department receives a project funding in accordance with the amount of persons bound in workshops which is ought to be used for desired projects.

During the workshops, initial parameters and information about KM conditions are surveyed systematically. These form the foundation for analyses about efficient knowledge flow within the company. Thus, the measure refers to the third phase of the GPO-WM called *Assessment of KM Conditions*.

Staff motivation is realized through an indirect financial reward at the department level. The individual motivation is realized if the department uses the project funding for the realization of individual project desires. Therefore, the measures addresses the *Skills & Motivation* design field.

Critical appraisal: The GPO-WM builds on the current process models of a company's business processes. Therefore, organizational elements, the processual behavior, and functional elements are addressed implicitly by the underlying process modeling language. Since requirements about the explicit modeling of relevant modeling entities have not been specified any further, modeling language criteria from the business process domain cannot be satisfied.

Although the specific forms and types of knowledge are not specified any further at the GPO-WM, the management of knowledge flows is enabled with GPO-WM. The concrete operationalization depends on the choice of the underlying knowledge process modeling language. Therefore, modeling language criteria from the KM domain are addressed by convention on an abstract level.

Neither the simulation of knowledge processes or business processes nor the use of ANN models are considered by the GPO-WM. Therefore, it is not prepared for dealing with simulation and ANN mechanisms and the corresponding modeling language criteria are not satisfied at all. Furthermore, relevant simulation and ANN mechanisms are not issued for use in tools, which is the reason for the poor performance of corresponding tool criteria. The use of tools only focus on criteria for the modeling of knowledge processes, their operationalization, and the knowledge

evaluation on an abstract conceptual level as concrete mechanisms are not provided by GPO-WM.

The GPO-WM Model enables a theoretically founded access to an organizational KM which makes it suitable as an interface for the CoNM to an organizational operation level. As the GPO-WM does not provide a technical foundation, the model's components can further serve as building blocks for a neuronal task realization. This must be based on a combination with concepts providing a technical foundation so that the dealing with simulations and neuronal mechanisms are brought together with KM concepts.

2.2.2.3 Potsdam KM Model

The Potsdam Knowledge Management Model is a management approach which enables operation on a common understanding of the terms “knowledge”, “knowledge management,” and the corresponding tasks relevant for adequate KM (Gronau, 2009, p. 55). As a processual KM approach, it enables the realization of efficient knowledge utilization within the company’s business process models and the creation of knowledge process models according to Def. 14.

Description: In the form of a framework, the Potsdam KM Model considers three ranges to characterize the measures of KM (Gronau, 2009, p. 50). First, a differentiation by the *organizational range* focuses on structural elements, such as measures affecting individuals, single organizations, or a combination of organizations. Second, a differentiation by the *procedural range* issues the management of a certain system, such as a system of activities, a system of processes, or a system of networks. Third, a differentiation by the *personnel range* enables the characterization of measures affecting a certain role, such as managers, knowledge managers, or knowledge workers.

Although any process modeling language can be used to visualize knowledge process models being affected by this management approach, the Potsdam KM Model is historically linked with the KMDL (introduced in section 2.2.3.3) because of their joint origin.

Side facts: The Potsdam KM Model was initially developed by Prof. Norbert Gronau and the *Know Learn Educate Group* of the University of Potsdam in 2009 as a basic framework for adequate knowledge management (Gronau, 2009). Since then, various tools and publications have been developed by the community which can be connected to the framework and can be interpreted as extensions. Accordingly, one can find modeling approaches, software tools, and concrete management mechanisms that do not change the basic concept. Currently, it is developed by

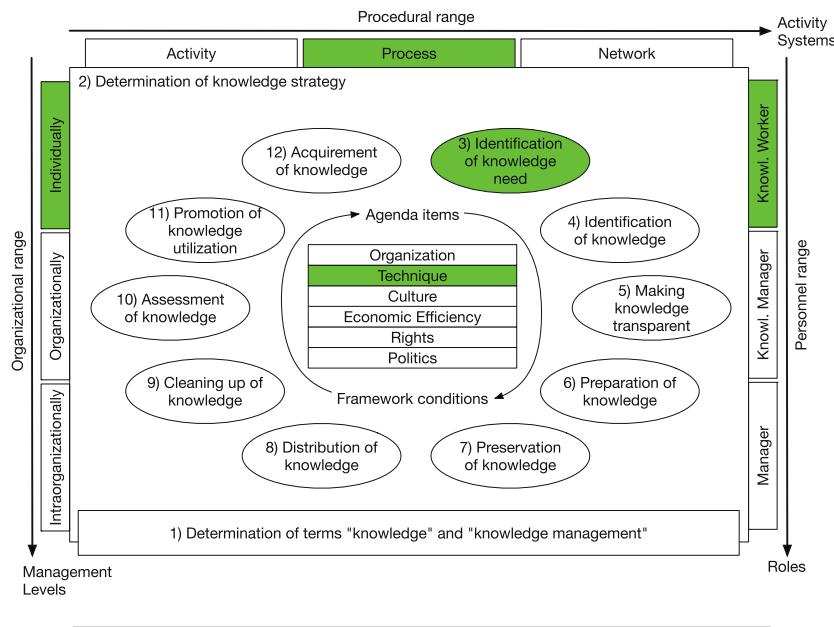


Figure 2.22 Potsdam Knowledge Management Model (highlighted elements issued in example)

the Department of *Business Informatics, especially Processes and Systems* of the University of Potsdam, headed by Prof. Gronau. As the most basic version, the following refers to the framework version of 2009.

Model: The components of the Potsdam KM Model are presented in Fig. 2.22 and described thereafter according to Gronau (2009). By the provision of building blocks for an adequate KM, any measure can be characterized by the three ranges introduced. The corresponding three ranges of a KM measure can be found at the outer edges of the visualization.

Further building blocks can be found in the figure center. Here, one can see twelve categories of measures that are demanded by an adequate KM. The numbers visualized refer to the following explanations and do not represent a sequential order.

1. *Determination of terms:* A common understanding of KM-relevant terms is established so that the whole organization is able to communicate efficiently.

This refers, for instance, to the terms “knowledge” and “knowledge management”.

2. *Determination of knowledge strategy:* Being part of the organizational strategy, the importance of knowledge is determined. This includes the establishment of an estimation about adequate efforts to obtain and extend organizational knowledge with respect to financial, technical, and organizational dimensions and the determination of strategic KM objectives. On the basis of this strategy, the subsequent ten operational KM tasks can be derived and concrete measures can be mapped to strategic KM objectives.
3. *Identification of knowledge need:* At an individual level, the kind of knowledge required for an effective process step realization is examined . This includes the characterization of knowledge by a description, time point, and location (Grum and Gronau, 2017).
4. *Identification of knowledge:* The current knowledge base of the organization is recognized by the examination and description of currently available knowledge on a personal, organizational, and inter-organizational basis.
5. *Making knowledge transparent:* Identified knowledge of internal and external knowledge carriers is visualized and made accessible by meta-information for any knowledge worker in an appropriate manner, such that the existence and location of the knowledge currently available becomes more transparent.
6. *Preparation of knowledge:* Knowledge is transformed to a form that allows efficient perception and use in current and future task realizations.
7. *Preservation of knowledge:* Individual and organizational knowledge is structured and backed up so that the efficient provision of knowledge in any business process and process instance is supported. Because of the use of context-based knowledge in other contexts, this is connected with knowledge decontextualization of the original context.
8. *Distribution of knowledge:* The distribution of knowledge is realized with respect to its efficient allocation to the location having a knowledge need either by direct management measures or by the self-organized distribution, which can be managed indirectly by the establishment of an adequate organization culture.
9. *Cleaning up of knowledge:* Faulty knowledge or knowledge that will not be demanded by the organization anymore is corrected, renewed, or deleted. Thus, its negative effects are avoided.
10. *Promotion of knowledge utilization:* Structural elements supporting knowledge use are established. This refers to the creation of new organizational structures, processes, and systems or the modification of existing ones.

11. *Assessment of knowledge:* The current knowledge base of an organization is assessed with regard to its value within the entire value chain. With this, it is assured that the objectives of the knowledge strategy are fulfilled.
12. *Acquisition of knowledge:* If a knowledge deficit is recognized, it is overcome either by internal procurement, e.g. by education or training, or by the external acquisition of experts, documents, consultants, etc.

Further, at the center of Fig. 2.22, one can see a visualization of *KM conditions* which need to be addressed by each knowledge management measure as since a knowledge management shall be considered as a social system (Gronau, 2009, p. 12). Only if measures design KM services as an offer for actors of the social system and these condition are fulfilled, measures will be successful since they are perceived to be optional and ideal for utilization. According to Kubicek, conditions refer to the following six (Kubicek, Westholm, and Wind, 2002; Kubicek, 2002).

1. *Organization:* Measures need to provide appropriate processes and an internal organization of communities in order to embed KM services in a company's structure.
2. *Technique:* Measures need to provide an appropriate infrastructure, software, and hardware, which allows the provision and utilization of KM services.
3. *Culture:* Measures need to address attitudes, qualifications, role understanding, and missions of actors of the social system, which all support the utilization of KM services and reduce barriers.
4. *Economic Efficiency:* Measures need to assure the beneficial, cost-, and time-effective KM service use for the social system actors considered.
5. *Rights:* Measures need to consider legal issues, such as property rights of third parties and conditions for governmental approvals about the provision of KM services. Further, they need to explain rights and duties for knowledge workers that are relevant for dealing with KM services.
6. *Politics:* Measures need to provide organizational resources and adjust legal circumstances that allow the use of KM services.

Example: An example measure characterization of the Potsdam KM Model is described in the following. It exemplifies the dimensions and components highlighted in green in Fig. 2.22.

Example (Potsdam Knowledge Management Example).

A measure will be realized that refers to the implementation of an online platform which allows knowledge workers to submit knowledge needs during the execution

of product development process steps. On behalf of hand-held devices, knowledge workers are asked to document the knowledge currently required.

Since the time point, location, and process step at which a knowledge need occurs is noted and this information will be reflected in the process models, the *process* attribute is affected in the *procedural range* dimension. As the platform was used for the identification of knowledge and its application in knowledge-based outcomes, the *activity* attribute was selected. The network was addressed if the knowledge transfer among networks of knowledge carriers was examined.

With this example measure, knowledge is primarily identified on an *individual* level, which is the reason for the corresponding attribute highlighting the *organizational range* dimension. In contrast, the survey of teams, departments (intraorganization), or interorganization-based circles could have issued collaborative knowledge needs.

Since a bottom-up procedure was selected to identify knowledge, the measure focuses on the *knowledge worker* attribute of the *personnel range* dimension. Therefore, other roles could have been asked to identify knowledge needs, such as the *manager* or *knowledge manager*, which would not lead to similar, adequate survey results.

The third measure category was selected because it is primarily identified at which process step which kind of knowledge is required. Although with this measure, the current knowledge base of the organization can be identified as well, the measure conducts a survey of knowledge needs.

Although any kind of condition needs to be met by the example, so far, the measure example primarily concerns the technical component of an online platform, which refers to the *technique* condition. Hence, the example measure is still incomplete and needs effort for completion.

Critical appraisal: As a management concept, the Potsdam KM Model particularly focuses on the management of knowledge and efficient processual anchoring. By the provision of concrete building blocks, the all-embracing and systematic KM realization is supported. Although modeling criteria are not considered explicitly by the management concept, the management of modeling items of the information and knowledge perspective, as well as items of the business process context and process domain (behavior, function, and organization perspective) are issued. Since KM measures prepare the development and use of modeling tools as well, modeling tool criteria are considered only indirectly.

Although the use of simulations has been issued for knowledge management (Gronau, 2009, p. 99, 100; Gronau and Grum, 2019), the management of simulation-

relevant items is not a consensus in KM, and the Potsdam KM Model does not concern simulation perspective items or simulation tools. Further, items from an ANN domain or concepts to manage knowledge by biological principles similar to the brain are not considered at all; the concept is realized on behalf of humans only. Thus, modeling criteria issuing NN and ANN as well as ANN tool criteria are not fulfilled either.

Based on a consensus of twelve KM concepts and overcoming their weaknesses of a missing differentiation of roles and measure ranges (Gronau, 2009, p.37–42), the Potsdam KM Model enables a theoretically founded access to organizational KM. Thus, it is attractive as interface for the CoNM to an organizational operation level. Because of a missing technical foundation of the Potsdam KM Model, the model's components can further serve as building blocks for a neuronal task realization.

2.2.2.4 Critical Appraisal of KMCs

Considering the aforementioned KM concepts and its interim conclusions, it seems that today's KM approaches show numerous and comparable aspects. However, they are still in development stage and do not consider simulation and ANN techniques:

- Only some KM approaches realize a process-oriented KM (e.g. Promote, GPO-WM and the Potsdam KM Model). This is essential as the processual anchoring of knowledge of the current, human-based KM shall be compared or brought together with the algorithmic, machine-based KM of ANN.
- None of the presented KM approaches considers neuronal processes and neuronal modeling objects and therefore, a management of ANN objects with currently available KM activities is missing.
- Further, as none of the presented KM approaches considers neuronal processes, no prominent representative has considered biological principles of brains in the knowledge organization and its economic contexts.
- Only some KM approaches consider *KM conditions* (e.g. GPO-WM's design fields or Potsdam KM Model's KM conditions) which supports the realization of an effective KM being integrated perfectly within a company's structure. This is essential as the company's workflows shall be optimized by neuronal mechanisms.
- Only some KM approaches consider a methodological proceeding from the beginning till continuous KM realization (e.g. Promote and GPO-WM). As the CoNM intends to enable human-based KM currently known with machine-based KM activities, this is essential.

- Only some KM approaches consider a systematic characterization of KM measures with regard to its ranges. Only the Potsdam KM Model distinguishes an organizational, personnel, and procedural range.
- Only some KM approaches address processual KM activities. While the GPO-WM provides very simple processual KM activities around a closed circle, the Potsdam KM Model provides 11 KM activities. As a higher number of activities brings more potential for the consideration on a neuronal level, the Potsdam KM Model's activities are ideal for the CoNM. Since this increases the complexity as well, the the GPO-WM is preferable because of its simplicity.

Considering these points, it becomes clear that a processual KM approach based on neuronal processes is not available yet. A research gap becomes visible here.

A synthesis of any KM approach presented here, enriched with simulation and ANN-required concepts and mechanisms, will support a broad acceptance of the CoNM in corresponding research communities. Further, a synthesis ensures its broad application possibilities. The idea for a 'holistic' synthesis efficiently represents all KM conditions, and an adequate proceeding as well as KM activities applicable in all modeling situations (organizational or neuronal level) must be avoided as they lead to KM approaches that are impractical and reduce the ease of use for any realistic setting. Considering a multiplicity of possible KM goals and objectives, this is challenging.

Therefore, for the construction of a CoNM, a compromise must be identified between holism of management items required and practicability. In principle, a practical and holistic synthesis must address the following things:

- It must broadly consider different kinds of processual KM activities (Gronau, 2009).
- According to Heisig (2002, p.53), it must support the construction of closed cycles of operational KM activities.
- Based on Heisig and Gronau, conditional KM aspects must be addressed (Heisig, 1999; Heisig, 2000; Heisig, 2002; Gronau, 2009).
- Following Gronau, it must consider various ranges (organizational, procedural and personnel ranges) so that social dimensions of knowledge are addressed adequately (Gronau, 2009, p. 50).
- It must provide a methodological proceeding (Hinkelmann, Karagiannis, and Telesko, 2002, p. 74; Heisig, 2002, p. 59).
- ANN and simulation techniques must be harmonized with the understanding of the KM approach and its corresponding knowledge modeling language.

The creation of a CoNM will therefore consider the aforementioned points and will be based on a practical synthesis of processual KM presented here.

2.2.3 Knowledge Modeling Languages

Considering the challenge to specify and communicate a modeling concept that allows the examination of knowledge transfers on neuronal levels and creates an interpretation of measurements of its activations, the modeling of tacit and explicit knowledge as well as its creation, individual transfer, and conditions of application require a documentation. Since this is connected with changes over time, the processual knowledge modeling is attractive as follows.

Generally, knowledge transfers can be modeled with the help of various modeling approaches. An overview and a comparison with regard to their capacity to represent knowledge can be found in literature (Remus, 2002; Sultanow et al., 2012). Relevant aspects of knowledge modeling are depicted in the following.

SECI-based Modeling: The operationalization of knowledge transfers can be realized using the concept of knowledge conversion (Nonaka and Takeuchi, 1995; Gronau and Grum, 2019) in which an instance of knowledge transfer can be considered over time. (section 2.2.1.3). SECI as a concept comprises four kinds of conversions which form the acronym for its title: Socialization, Externalization, Combination and Internalization. Numerous definitions and interpretations of each conversion type can be found in literature (Nonaka and Takeuchi, 1995; Gray and Densten, 2005; Desouza and Awazu, 2006). The most prominent interpretation considers conversion as the interplay of different forms of knowledge during the realization of an activity (Nonaka and Takeuchi, 1995, p. 61). Some knowledge modeling languages are directly based on the SECI-concept, for e.g. KMDL (Gronau, 2012), such that the duration and quality of each conversion type, for example, can be observed perfectly (Gronau and Grum, 2019).

Perspective-wise Modeling: Particularly, a view-based separation and the combination of different modeling perspectives about the same object of reflection in one model enables the purpose-related visualization, which is essential for the construction of a CoNM. As the sequential description of processes (processual perspective) is separated from the process of knowledge creation, transfer, and application (knowledge perspective), the latter can be embedded in the process description which simplifies the understanding because of clear process models (Gronau and Grum, 2019). Further, this allows the specification of knowledge transfers within

different knowledge transfer settings for each process step and supports the CoNM construction intended.

Position-based Modeling: For a successful process step realization, tangible objects, such as material or products, as well as intangible objects, such as knowledge, need to be present. Since these are related to a unique spatial positioning, process modeling approaches as well as knowledge process modeling approaches need to consider an object's position within a real-world coordinate system (Grum and Gronau, 2017).

Critical Appraisal: Considering the knowledge understanding stated in Def. 13, including the processual understanding of knowledge transfers (Def. 14) and the unique character of business processes (Def. 4), this research defines ordinary process modeling languages in accordance with (Def. 8) the considered knowledge process modeling languages as follows:

Definition 15 (Knowledge Process Modeling Language).

A knowledge process modeling language (synonym for knowledge process modeling notation) is a, not necessarily but mostly, graphical notation satisfying more or less sophisticated syntactic rules to be applied by a model creator to support the construction of a model creator's reality and to create a representation of its subjective or objective reality about processes and the use of knowledge, the so-called knowledge process model, within the creator-specific subjective or objective model environment on arbitrary levels of abstractions and granularities of deep structures.

As the definition does not exclude specific process models, it includes the specialization of domain-specific process models, such as business process model or process models for (artificial) neuronal networks. Here, each domain demands individual process modeling elements in coherence with the most basic and abstract meta-model of deep structures available (see Fig. 2.2). Although this relation is not visible when considering a specific knowledge process model example and most knowledge process modeling notations do not satisfy this formalization adequately, only addressing very specialized issues and only a small selection of common modeling perspectives (cf. section 2.1.2), this relation is essential as it holds the (modeling) world together.

According to the *modeling definition* (Def. 5), and similar to the use of process modeling languages (section 2.1.3), during knowledge process modeling, knowledge is considered by the model creator as follows: First, knowledge is considered to abstract from a sophisticated reality by the model creator (see *forgetful mapping*

of Fig. 2.1). Second, knowledge is considered to complement models with attributes (see *extension mapping* of Fig. 2.1). Third, knowledge is considered by the model creator to efficiently and appropriately express representations on behalf of the corresponding knowledge process modeling language and its conventions. Since these kinds of knowledge uses are available within the model constructed in a tacit manner, the subjective and the hardly comprehensible view of the model creator leads to a complicated tension between original creator-model relationship depicted in Fig. 2.5.

The term *mostly* in the definition implies that knowledge processes can be carried out by a combination of *graphical modeling items*, but can also refer to *mathematical expressions* or a corresponding representation as *script language* in the sense of computer-aided software engineering (CASE) as well (Case, 1985; Bergin, 1993; Bucci et al., 1994; Lang and Duggan, 2001). The common denominator here is the use of language constructs, such as *words* and *grammatics*. By following transformation rules, the models following different notations or representation forms can be *translated* in the one or other language. As the corresponding two models are well formalized, this translation can be carried out without information loss or information writing which refers to the same level of *information entropy*.

Selection of representatives: As the a great number of processual knowledge modeling notations can be found in literature, the selection of adequate representatives must be addressed. Attempts searching for best candidates refer to the following e.g.: Remus, 2002; Sultanow et al., 2012; Grum and Gronau, 2017. The contribution therefore selects knowledge process modeling languages according to the following criteria.

1. Modeling languages having a broad market distribution determined by market research studies (Remus, 2002; Sultanow et al., 2012; Grum and Gronau, 2017).
2. Modeling languages having extensive acceptance in the modeling community
3. Modeling languages applied in at least one scientific publication

Adequate examination level: To satisfy requirements for an adequate management of objects of investigation (OoI) in the terms of SLR (Levy and Ellis, 2006) which corresponds to the application of Bloom's taxonomy (Bloom et al., 1956), each knowledge modeling language is described by the following categories:

- The short *abstraction* which serves as a sort of introduction to modeling languages on the same abstract level
- Its individual *description* which characterizes the individual OoI

- *Side facts* issuing the OoI-specific development and historic evolution.
- The provision of a *meta-model* visualizes modeling items and its underlying syntax. This is completed by *modeling conventions* clarifying how the meta-model of the OoI is applied so that concrete process models are constructed.
- An *example* visualizes the application and underlines modeling strengths of the OoI.
- Finally, the *critical appraisal* evaluates whether and how the individual modeling language is suited to stand as foundation for the CoNM.

Thus, according to Bloom's Taxonomy, categories considered for any knowledge modeling language correspond to the levels of *knowing* and *comprehending* if OoIs are described and side facts are presented, *applying* as a meta-model is generated and an example is presented, *synthesizing* as a common level of understanding is worked out as well as *analyzing* and *evaluating* as common analysis criteria are issued and the knowledge modeling languages are considered with respect to the CoNM.

2.2.3.1 Promote

The **Promote** modeling language is a graphical modeling language which focuses on the modeling of different kinds of KM processes (Karagiannis and Telesko, 2000; Woitsch, Hrgovic, and Buchmann, 2012) and thus enables the processual anchoring of knowledge objects within a company's set of business process models. In this manner, it allows the construction of knowledge process models in accordance with Def. 14. Hence, it is suited for the modeling of the behavioral perspective of a system and considers objects of the knowledge flow-oriented perspective (section 2.1.2.1).

Description: KM processes focused by the Promote modeling language refer to the following three kinds (Karagiannis and Telesko, 2000). First, *acquisition* processes intend to integrate new knowledge in the organizational memory. This considers the direct, guided, and automatic acquisition as well as interactions of the knowledge producer and the knowledge consumer. Second, *search and retrieval* processes intend to access knowledge from the organizational memory. This issues query by full-text search and entity-based search as well as access on behalf of knowledge navigation assistants. Further, adequate knowledge selection is supported and the use of information agents is enabled. Third, *maintenance* processes realize the management of the organizational memory and include e.g. processes for archiving, validation, feedback surveying, and the transformation between knowledge formats.

The modeling on behalf of Promote is realized on the basis of the interplay of two views: a view for processes and a view for knowledge activities. Each is presented in the following.

First, the *Business Process View* visualizes the structure of a process and *tasks* related to it. *Control flows* are visualized by arrows. These connect tasks and shapes representing *Boolean operators*. On behalf of them, decisions and parallel task sequences can be modeled so that time-dependent relations are visualized. Each decision is characterized by the use of *descriptions*. Each task or a collection of tasks represented by *process interfaces* is associated by a *denomination*. Tasks are optionally characterized by an *indicator for knowledge-intensive activities*.

Second, the *Knowledge Management Process View* shows operationalizations of knowledge flows identified (Hinkelmann, Karagiannis, and Telesko, 2002, p. 80). Following the same visualization elements of the Business Process View, here, *knowledge activities* are considered instead of tasks. As some activities are mapped to tasks of the Business Process View, the corresponding processual context initiates processes modeled by the Knowledge Management Process View (Hinkelmann, Karagiannis, and Telesko, 2002, p. 82).

Side facts: Besides the processual knowledge management approach (section 2.2.2.1) and the corresponding software (introduced in section 2.6.1.3), the Promote modeling language was developed by Prof. D. Karagiannis in the EU project having the same name (Karagiannis and Telesko, 2000; O'Brien, 2010). As the software development was overtaken by the *BOC Information Technologies Consulting AG*, the modeling language development was primarily overtaken as well.

Meta-model: Fig. 2.23 provides the general *Promote business process meta-model*. It clarifies the interplay of modeling objects of different views.

Considering the presented general business process model of the modeling language of Promote, Promote-based models are constructed by the following Promote design rules (derived from models of Karagiannis, 1995; Karagiannis, Junginger, and Strobl, 1996; Karagiannis and Woitsch, 2002; Hinkelmann, Karagiannis, and Telesko, 2002; Karagiannis and Telesko, 2000; Spies and Trojan, 2007; O'Brien, 2010):

1. Promote's core nodes refer to *tasks* and *activities* and logical *connectors*.
2. The task's and activity's *denomination* should reflect its time-consuming perspective of the accomplished process step.
3. Connectors are represented by a circle as well as its corresponding denomination and can be split into upper and lower parts so that one can differentiate between

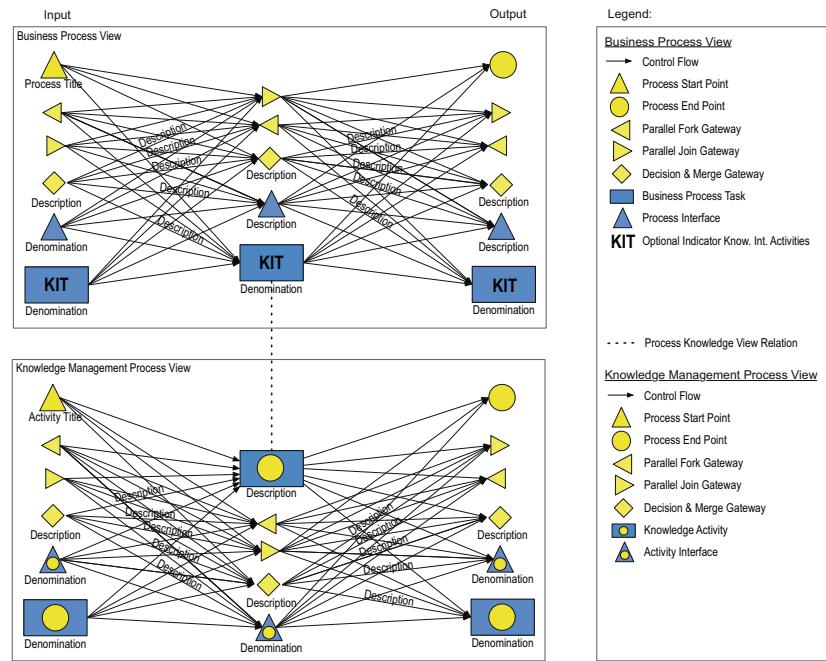


Figure 2.23 Generic Promote business process meta-model (synthesized from literature)

incoming and outgoing rules. Here, one can find the *AND rules* and the *OR rules*. As the latter are associated with descriptions, here, one can realize *OR rules* and *XOR rules* by language.

4. Promote-based models start with a *start point* and end with an *end point* so that one can define when a business process is intended to begin and finish.
5. Promote-based business process models at least comprise one atomic task or a composition of further tasks.
6. Promote-based knowledge management process models at least consist of one atomic activity or a composition of further activities. A pictogram inside the circle presents a visualization of the corresponding activity.
7. Edges connect two tasks or activities corresponding to the sequence of action; thus, they are directed.
8. Each task has only one incoming and one outgoing edge.

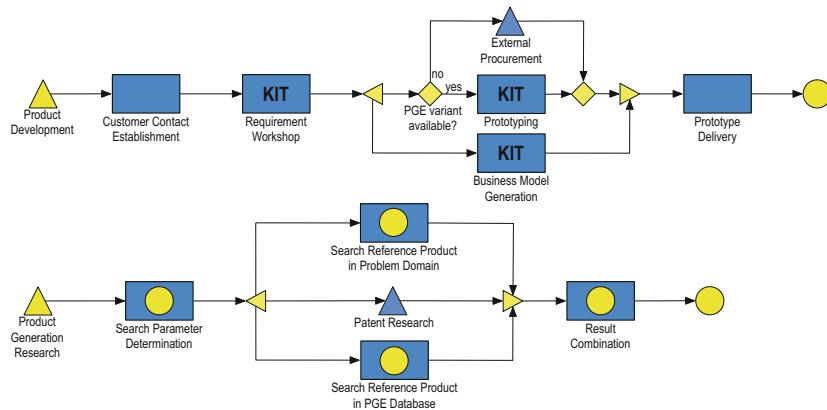


Figure 2.24 Promote-based process model example (created to show the strength of the object of investigation)

Example: An example for a Promote model is visualized in Fig. 2.24. It shows the product development inspired by the product generation engineering (Albers et al., 2014) and thus clarifies the application of Promote conventions.

While the model at the top shows the business process called “Product Development”, the model below shows the KM process known as “Product Generation Research.” The latter might be triggered by the task “Requirement Workshop” or by the task called “Prototyping.” Since both tasks demand knowledge about comparable *reference products*, knowledge flows can be identified here within the same process. The first might derive requirements in dependence of reference products considered and the second might derive variations dependent on the reference product selected. Since both types of processes are modeled, the interplay of business processes, KM processes and knowledge flows becomes clear. Since the operationalized KM processes can be automated by systems and workflow engines, the effective realization of different kinds of KM processes is enabled.

Critical appraisal: The modeling language of Promote focuses on the modeling of knowledge flows and enables the management of knowledge. Hence, it can be characterized as knowledge process modeling language. It provides basic items for the processual description of the behavior of a system. This refers to simple boolean operators and the realization of processes and process interfaces. No other sophisticated modeling item is provided such that only very basic modeling

language criteria from the business processes context are satisfied. Organizational entities or entities required for the modeling of simulation issues are not considered so that the corresponding modeling items are not satisfied at all.

At least knowledge-intensive process steps are characterized so that knowledge flows are operationalized by Promote's knowledge view. Since these are not differentiated from the control flow, Promote does not consider any further knowledge construction. Thus, conversions are not considered adequately here, and the corresponding modeling language criteria of the knowledge perspective are not satisfied completely.

Further, since neuronal mechanisms or simulation mechanisms are not issued at all, no modeling language criteria from the ANN domain or the use of ANN models in simulations is satisfied. Thus, the corresponding tool criteria are disregarded as well.

Promote only enables very basic mechanisms for modeling tools. As modeling conventions are not provided on a formalized level, a modeling tool can refer to a semi-formal graphical modeling than a valid model construction including a collaborative, AR-based process modeling. Thus, the corresponding modeling language and tool criteria are not satisfied adequately. Since Promote focuses on knowledge transfers and incorporates the context of business processes, it is suitable for a combination with ANN tools as a basis for the construction of a CoNM. Because of a missing possibility to model knowledge construction, this is only possible with either circumstantial and laborious modeling language expansions or with a combination of further modeling language components.

2.2.3.2 GPO-WM

The GPO-WM is a graphical modeling language. It enables the processual anchoring of knowledge objects within a company's set of business process models, allowing the construction of knowledge process models. Hence, it is suitable for modeling the behavioral perspective of a system and considers the objects of the knowledge flow-oriented perspective (section 2.1.2.1).

Description: KM processes considered by the GPO-WM modeling language refer to the following four kinds (Heisig, 2002, p.55). First, *generate* processes intend to create new knowledge relevant for the successful realization of business processes. Second, *store* processes intend to record knowledge in organizational memory. Third, *distribute* processes support the provision of knowledge among process participants so that a business process task is realized efficiently. Fourth, *apply* processes focus on the use of knowledge. These four kinds of processes have an operational relevance: If possible, they need to create a closed cycle so that the orga-

nizational knowledge base is expanded efficiently (Heisig, 2002; Gronau, 2009). Two further kinds of processes refer to processes having a strategic relevance. First, *knowledge objective determination* processes, ensure the target-oriented realization of processes. Second, *knowledge identification* processes focus on the search of knowledge being constructed in business processes.

The modeling on behalf of GPO-WM is realized on the basis of two views: one view for business processes and another view for information. The view for business process models shows objects of an enterprise including their interdependencies. Thus, business process models visualize the processual realization of activities including their interaction with objects of the types *order*, *resource*, and *product*. Accordingly, the analyses with closed cycles of knowledge activities are carried out. The view called “information model” further specifies quantities and relations of these three kinds of objects in the form of class trees.

Side facts: The GPO-WM modeling language was developed by Prof. Peter Heisig from the *Competence Center* of the Fraunhofer IPK in Berlin (Heisig, 2002). It is built on the foundation of the IUM, which is the Fraunhofer method for integrated organization modeling (German: Methode der Integrierten Unternehmensmodellierung, short: IUM) and was developed to map, describe, analyze, and design organizational processes (Süssenguth, 1991; Spur, Mertins, and Jochem, 1993; Schwermer, 1998). Going beyond the IUM, the GPO-WM further provides knowledge elements which are considered to be one of its object types either as *resources* or as *products*. The modeling language can be used in accordance with the GPO-WM concept (section 2.2.2.2) and the software tool called Mo²Go.

Meta-model: Fig. 2.25 provides the general *GPO-WM business process meta-model*. It clarifies the interplay of modeling objects within the two views.

Considering the presented general business process model of the modeling language of GPO-WM (Heisig, 2002), GPO-WM-based models are constructed by design rules not formally specified and derived from the IUM meta-model (Süssenguth, 1991; Spur, Mertins, and Jochem, 1993; Schwermer, 1998) and GPO-WM model examples:

1. GPO-WM's core nodes refer to *actions*, *orders*, *resources*, and *logical connectors*.
2. The action's *denomination* name should reflect the time-consuming perspective of the accomplished process step.

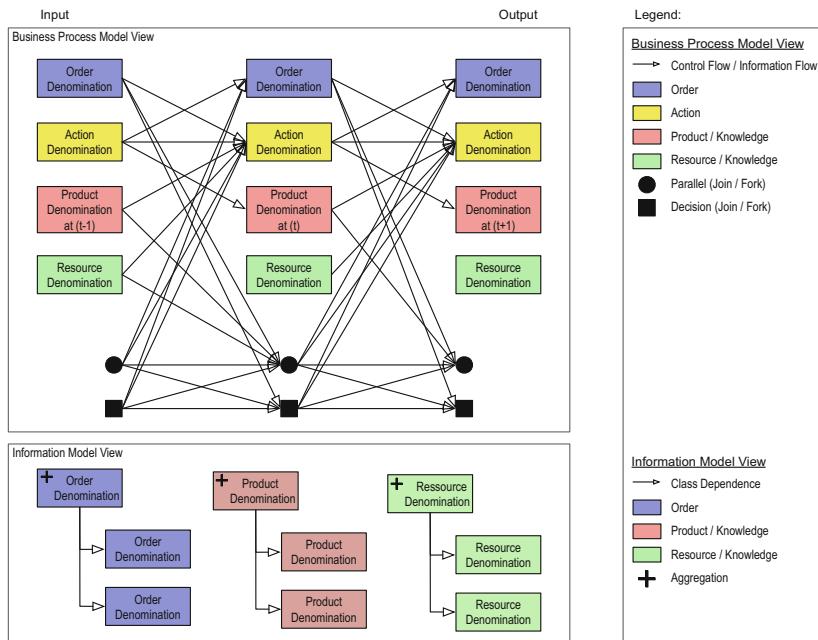


Figure 2.25 Generic GPO-WM business process meta-model (synthesized from literature)

3. Connectors are represented by black circles and rectangles and can split into upper and lower parts so that one can differentiate between incoming and outgoing rules. Here, one can find the *AND rules* (circles) and *XOR rules* (rectangles). Representations for the *OR rules* are not provided.
4. GPO-WM-based models start with a *product* or *order* as start point and end with an *order* or *product* as end point so that one can define when a business process is intended to begin and finish.
5. GPO-WM-based business process models at least consist of one atomic action or a composition of actions.
6. Edges connect two actions corresponding to the sequence of action, information flow, or knowledge flow and are represented by arrows having a blank arrow-head. Thus, they are directed. A separate representation for each edge is not differentiated.
7. Each action has only one incoming and one outgoing edge.

8. Class diagrams show a hierarchy of object types. Here, the very bottom object of the hierarchy does not seem extendable by the “+” symbol.

Example: An example for a GPO-WM model is visualized in Fig. 2.26. It shows a minimal production process and so clarifies the application of GPO-WM conventions.

On the left, one can see the business process model. It starts with the initial product components at time step $t = 0$ and ends with the final product at time step $t = 1$. In between, one can find a knowledge object, which in the example is from the IUM object type *product*. On the right, one can see the information model about the *resource* object types. It only shows entities used in the process model. Regarding the intention to construct closed cycles of knowledge activities, the process model shows the following cycles: Knowledge is generated during the action called “discuss production planning”. It is stored during “document lessons learned”. As it is distributed on behalf of a system enabling the search for reference cases, the corresponding knowledge about similar reference cases is applied in the action called “research reference orders.” Besides the circle described, further elements can be found in the example presented.

Critical appraisal: The GPO-WM modeling language focuses on the modeling of knowledge flows and enables the management of knowledge. Hence, it can be characterized as knowledge process modeling language. It does not merely provide basic items for the processual description of the behavior of a system. These refer to simple Boolean operators and the realization of processes and process interfaces. As entities describing material products and orders are provided, this includes the processual description on behalf of the business process model and further includes the quantitative description on behalf of information models; thus, the construction of more sophisticated models is enabled. Hence, modeling language criteria from the business processes context are satisfied. Organizational entities are considered adequately by the resource object type. Thus, the corresponding modeling items are satisfied.

At least knowledge-intensive process steps can be characterized by the objectification as resource or product object type. Thus, knowledge flows can be operationalized by the GPO-WM’s business process model view. Since these are not separated from the control flow, GPO-WM does not enable the modeling of knowledge construction. Here, conversions are not considered adequately, and the corresponding modeling language criteria of the knowledge perspective are not satisfied completely.

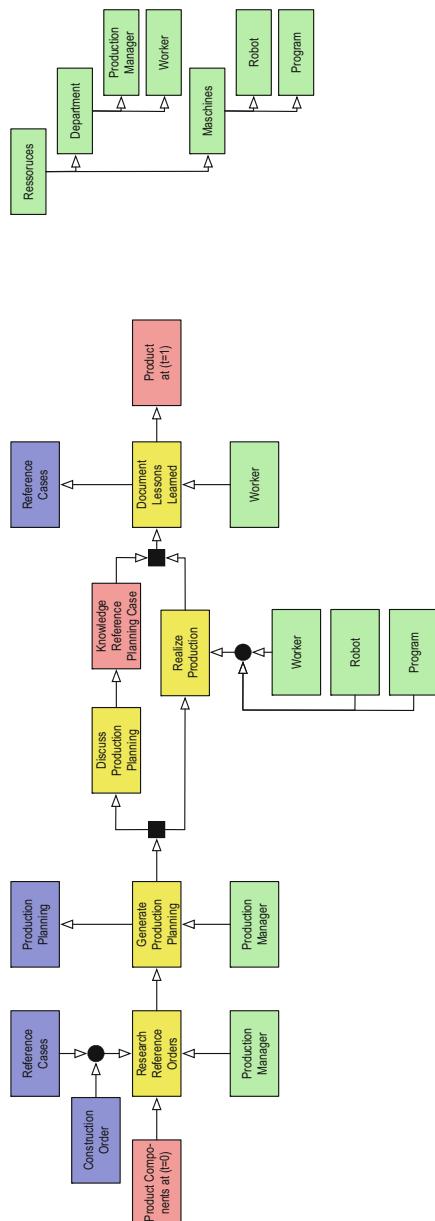


Figure 2-26 GPO-WM-based process model example (created to show the strength of the object of investigation)

Further, no modeling language criteria from the ANN domain or the use of ANN models in simulations is satisfied since neuronal mechanisms or simulation mechanisms are not issued at all. Thus, the corresponding tool criteria are disregarded as well.

GPO-WM only enables very basic mechanisms for modeling tools. As modeling conventions are not provided on a formalized level, a modeling tool can instead refer to a semi-formal graphical modeling than a valid model construction being used in simulations. A collaborative, AR-based process modeling is not issued at all. Therefore, the corresponding modeling language and tool criteria are not satisfied adequately. As the GPO-WM process modeling language at least enables the modeling of knowledge transfers and realizes the processual anchoring within the context of business processes, it is suited for a combination with ANN tools as a basis for the construction of a CoNM. Because of a missing possibility to model knowledge construction, this is only possible with either circumstantial and effort-intensive modeling language expansions or with a combination of further modeling language components.

2.2.3.3 KMDL

The Knowledge Modeling and Description Language (KMDL) is a graphical modeling language, which enables the processual anchoring of knowledge objects within a company's set of business process models and so allows the construction of knowledge process models. Hence, it is suited for the modeling of the behavioral perspective of a system and considers objects of the knowledge flow-oriented perspective (section 2.1.2.1).

Description: The modeling on behalf of the KMDL is realized on the basis of the interplay of three views: a view for processes, a view for activities, and a view for the communication. Each is presented in the following.

First, the *Process View* visualizes the structure of a process and *tasks* related to it. The *control flow* is visualized by arrows that connect tasks and *Boolean operators*. The latter allows the modeling of decisions and parallel task sequences so that time-dependent relations can be modeled. As *information systems* or persons having a specific *role* participate in the task realization, the corresponding shape is associated by a *membership connection*. Fig. 2.27 presents elements of the process view on its top.

Second, the *Activity View* visualizes the transfer of knowledge during the completion of a task modeled in the process view (Pogorzelska, 2012, p. 143). Fig. 2.27 presents elements of the activity view in the center. The *conversion* depicts knowledge transfers between explicit and tacit knowledge (Reber, 1989; Smith, 2001; Non-

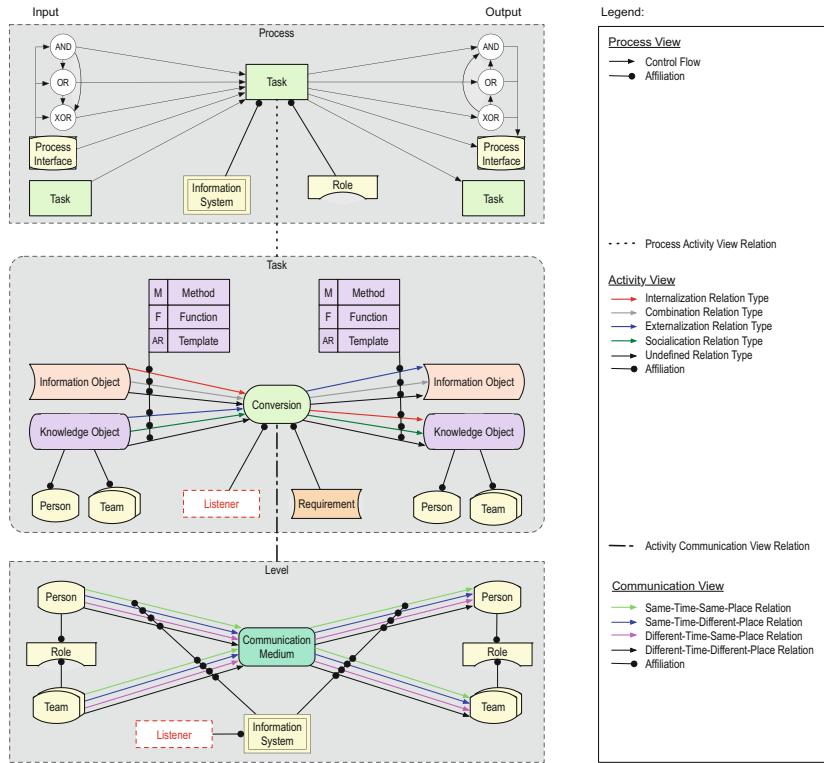


Figure 2.27 Generic KMDL business process meta-model (synthesized from literature)

aka and Takeuchi, 1995). So, the *knowledge flow* in the form of arrows connects *knowledge objects* and *information objects* as the input and output of a conversion. The different kinds of conversions defined by Nonaka and Takeuchi (see knowledge transfers of section 2.2.1.3) are visualized in different colors of the arrows. The more frequent a conversion that is carried out, the thicker the arrow visualizations. Since knowledge is specific to individuals, any knowledge object is associated with a *membership connection* and an individual representing *persons* or *teams*.

Third, the *Communication View* visualizes the communication during the completion of a task or rather activity on behalf of a *communication medium*. Fig. 2.27 presents its elements at the bottom. The relations visualized by arrows characterize

the communication process with regard to the place and time of the information sender and information receiver. The communication partners refer to *persons* or *teams* having a certain *role*. Whether the communication is planned or not is visualized by the degree of the arrow's dots. The more the dots used, the more planned is the communication process carried out. Further, it is modeled by a *listener* under which conditions an *information system* is used.

Side facts: The KMDL initially was developed by Prof. Norbert Gronau from the *Knowledge Management Project Group* of the University of Oldenburg in 2003 as semi-formal modeling concept (Gronau et al. 2003; Gronau, 2003; Gronau and Weber, 2004b). Since then, various KMDL variants and extensions have been developed by its community. These provide further modeling entities, business objects, organizational units, and further modeling views without changing the original KMDL concept (Fröming, 2009; Gronau, 2012; Gronau et al., 2016; Grum and Gronau, 2017). Currently, it is developed by the Department of *Business Informatics, especially Processes and Systems* of the University of Potsdam headed by Prof. Gronau. Since the most current version refers to the *KMDL* of vers. 2.2 (Gronau and Fröming, 2006, p. 105; Pogorzelska, 2012), the following considers the most modern version.

Meta-model: Fig. 2.27 provides the general *KMDL business process meta-model*. Here, it becomes clear how modeling objects of different views are integrated by task and conversion elements.

Considering the presented general business process model of the KMDL, KMDL-based process models are constructed by the following KMDL design rules (Pogorzelska, 2012, p. 113–129; Gronau, 2012; Grum and Gronau, 2017):

1. KMDL's core nodes refer to *tasks* and *conversions* which are sometimes called *activities*, logical *connectors*, and knowledge which refers to *knowledge objects* as well as *information objects*.
2. The activity's name should reflect its time-consuming perspective of the accomplished task or conversion.
3. Connectors are represented by a circle as well as its corresponding denomination and can split in into upper and lower parts so that one can differentiate between incoming and outgoing rules. Here, one can find the *AND rules*, the *OR rules*, and the *XOR rules*.
4. KMDL-based process models start and end with a *task* or *process interface* so that one can define when a business process is intended to start and finish.

5. KMDL-based process models at least consist of one atomic task or a composition of further tasks.
6. Edges connect two tasks corresponding to the sequence of action by arrows; so they are directed.
7. Each task has only one incoming and one outgoing edge.
8. KMDL-based process models relate *information systems* with the task they support by an *affiliation*.
9. KMDL-based process models relate *roles* with the task they support by an *affiliation*.

Further, the conventions are complemented by conventions for KMDL-based activity models (Pogorzelska, 2012, p. 113–129; Gronau, 2012; Grum and Gronau, 2017):

10. KMDL-based activity models start and end with a *knowledge object* or *information object* so that one can define when a knowledge transfer process is intended to begin and finish.
11. The knowledge name should reflect its unique characteristic as a point in time and is represented by a *knowledge object* or *information object*.
12. KMDL-based activity models at least comprise one conversion or a composition of further conversions.
13. Each conversion has at least one incoming and one outgoing edge visualizing the knowledge flow. This is directed and visualized by arrows colored with respect to one of the four basic relation types: internalization, externalization, socialization, or combination. The set of incoming and outgoing objects, the conversion object, and its edges is called *atomic conversion*.
14. Each conversion set with more than one incoming or outgoing edge, whose kind of relation type still can be identified, is called *complex conversion*. The corresponding edges are colored with respect to the four basic relation types.
15. Each conversion set with more than one incoming or outgoing edge, whose kind of relation type can not be identified anymore because multiple kinds of conversions are realized simultaneously, is called *undefined conversion*. Here, the undefined edge is colored in black. Edges that still can be identified are colored with respect to the four basic relation types.
16. Two conversions are connected indirectly by its *knowledge objects* and *information objects* as well as required edges corresponding to the sequence of action. Thus, they consist of the alternating elements of *conversions* and knowledge (*knowledge objects* and *information objects*).
17. KMDL-based activity models relate *requirements* or *listeners* with the activity they characterize by an *affiliation*.

18. KMDL-based activity models relate *methods*, *functions*, and *AR templates* with the edge they characterize by an *affiliation*.

Finally, the conventions include design rules for KMDL-based communication models (Pogorzelska, 2012, p. 113–129; Gronau, 2012; Grum and Gronau, 2017):

19. KMDL-based communication models start and end with a *person* or *team* so that one can define with whom a knowledge transfer process begins and finishes.
20. The *level* characterizes the place at which a communication occurs.
21. KMDL-based communication models at least consist one communication process requiring at least one *communication medium* or a composition of further communication processes.
22. Each *communication medium* has at least one incoming and one outgoing edge visualizing the communication flow. This is directed and visualized by arrows colored with respect to one of the four basic relation types: the *same-time-same-place* communication type, the *same-time-different-place* communication type, the *different-time-same-place* communication type, or the *different-time-different-place* communication type.
23. Two communication mediums are connected indirectly by its *persons* and *teams* as well as required edges corresponding to the sequence of action. Thus, they consist the alternating elements of *communication mediums* and communication partners.
24. KMDL-based communication models relate *roles* with the *person* or *team* to which they belong by an *affiliation*.
25. KMDL-based communication models relate *information systems* with the edge they support by an *affiliation*.

Example: An example model for a KMDL is visualized in Fig. 2.28. It shows the project acquisition process inspired with help of the three KMDL-specific views and thus clarifies the application of KMDL conventions.

On the very top of the figure, one can see the sequence of tasks required for the project acquisition on behalf of the process view. The process system to be modeled is delimited by the dashed system border. Irrelevant tasks are thus not considered in the process model presented. Since the task called *Team Evaluation* can be characterized as knowledge intensive, it can be detailed in the activity view. An example model can be found in the figure center. As conversions are realized repetitively, the clear temporal order of the process view is broken down in the activity view. The bottom of the figure presents the communication view which places the activity called *evaluation* and *documentation* within the real world. As

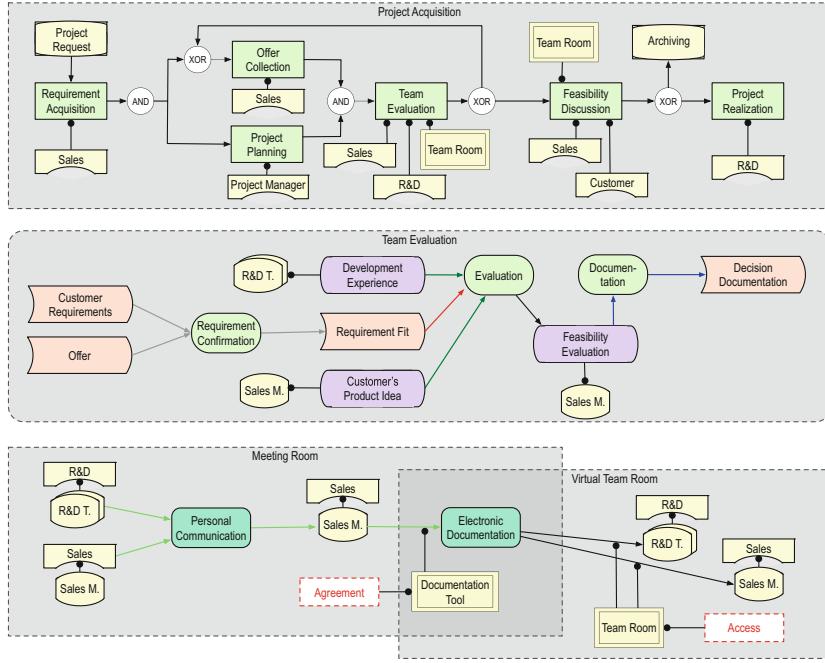


Figure 2.28 KMDL-based process model example (created to show the strength of the object of investigation)

the model shows, the team-based evaluation is carried out in the *meeting room*. As an *agreement* is identified, it is documented on behalf of the *documentation tool* within a *virtual team room*. Thus, the *R&D team* and the *salesman* can access the *decision documentation* at any place and any time.

Considering this example, the efficient and powerful interplay of the three KMDL views becomes clear. As this coupling is rather loose and semi-formal than strictly formalized, the process constructor is demanded to bring in creativity and realize a coupling on behalf of interpretations. Therefore, the application of concepts presented in process simulations require high efforts although approaches for this can be identified in the literature (Fröming, 2009; Gronau and Grum, 2019).

Critical appraisal: The KMDL has been proven in practice, and many examples can be found in business and academic contexts (Ammon et al., 2009; Neumann, 2015; Laukemann, Binz, and Roth, 2017). It clearly shows strengths in visibility as well as its informational perspective and has proved its capabilities compared to twelve other modeling approaches (Sultanow et al., 2012; Grum and Gronau, 2017). Although the concept of the KMDL does not consider modeling objects from the simulation domain and corresponding modeling criteria cannot be satisfied, the KMDL clearly shows strengths in the modeling of business processes including the processual anchoring of knowledge. Thus, apart from the satisfying fulfillment of common modeling criteria of the business process domain which requires criteria from the behavior, function, and organization perspective, criteria from the knowledge domain can be satisfied. As the *information object* must be differentiated by *data objects*, an adequate data processing cannot be realized by KMDL-based models and corresponding criteria are not satisfied.

Although a formalization of the KMDL is not available, approaches of the KMDL can be mapped to EPC-based models (Fröming, 2009). Since these have been formalized, the KMDL is prepared for use in software tools. Using this, a simple rule checking can be carried out by tools and thus enables comfortable model construction. As a great collection of analysis concepts focuses on the generation of reports, further, a significant collection of optimization concepts focuses on the identification of process patterns and unconstrained improvement (Gronau, 2012). The KMDL further provides a validated research method (Gronau and Weber, 2004a; Ranjbarfard et al., 2013). It is best prepared for tool-based and guided model construction. An example can be found in the tool called Modelangelo (section 2.6.1.4). Because of the following three reasons, the concept of KMDL is very suited for the construction of a CoNM from a process modeling side: First, the modeling concept of the KMDL considers a positioning within the real world by the communication view; second, modeling concepts on the basis of the KMDL extend positions in the real world space on behalf of coordinates (Grum and Gronau, 2017) and third, even a first prototype called *Augmentor* enables an AR-based modeling (Grum and Gronau, 2018b). Since the CoNM requires the consideration of spatial relations, a fundamental CoNM requirement can be satisfied with the KMDL.

Further, the most attractive advantage of the KMDL to other knowledge process modeling languages is that it enables the modeling of knowledge constructions, this is separated by a proper view, the activity view.

Unfortunately, no modeling item from the ANN domain is provided by the KMDL. Further, concepts required for dealing with artificial and non-artificial neuronal networks are not considered. Thus, the corresponding modeling language and tool criteria are not satisfied at all. Since the KMDL provides a focus on knowl-

edge transfers and brings the context of business processes in, it is suited for a combination with ANN tools as a basis for the construction of a CoNM.

2.2.3.4 Critical Appraisal of KMLs

Considering the aforementioned knowledge modeling approaches and its interim conclusions, it seems that today's processual knowledge modeling approaches show numerous aspects which can be found in process modeling taxonomies (section 2.1.2), but they are still developing and do not consider ANN techniques:

- None of presented knowledge process modeling languages consider neuronal processes, neuronal modeling objects and therefore, an integration of ANN objects with currently available process modeling perspectives is missing.
- None of the presented knowledge process modeling languages consider an *EA concept* compared to the OoI of ARIS (section 2.1.2.3.1), which supports the construction of consistent process models and knowledge process models being integrated perfectly within a company's structure. This is essential as workflows shall be optimized by neuronal mechanisms.
- None of the presented knowledge process modeling languages consider phases dealing with neuronal libraries, neuronal tools, neuronal methods, or neuronal concepts. An integration with currently available concepts, tools, or libraries is therefore missing as well.

Considering the aforementioned points mentioned, a research gap becomes visible.

A synthesis of any knowledge modeling approach presented here enriched with ANN modeling objects and required concepts and mechanisms will support a broad acceptance of the CoNM in corresponding research communities. Further, a synthesis ensures broad application possibilities. Thus, following the idea for a 'holistic' synthesis introduced in section 2.1.3.5, which efficiently represents all modeling perspectives as well as objects and is applicable in all modeling situations, the compromise must be identified between the holism of modeling items required and practicability. Thus, the practical and holistic synthesis must address the following things:

- It must consider the modeling of business processes as well as the modeling of knowledge flows. This includes knowledge transfers (Reber, 1989; Karagiannis and Telesko, 2000; Smith, 2001; Heisig, 2002; Gronau, 2012; Woitsch, Hrgovic, and Buchmann, 2012).

- In accordance with Gronau, it must address the construction of knowledge. This includes different forms of knowledge generation, such as internalization, socialization, combination, and externalization (Nonaka and Takeuchi, 1995; Gronau 2012).
- Following Grum and Gronau, it must enable the spatial positioning of modeling entities (Grum and Gronau, 2017).
- ANN techniques must be harmonized with the understanding of the business process modeling and knowledge modeling.

The creation of a CoNM will therefore consider these points and will be based on a practical synthesis of the knowledge process modeling languages presented here.

2.2.4 Conclusion About KMCs and KMLs

The conclusion about knowledge modeling is based on the following. First, knowledge management concepts (KMCs) are appraised. Then, knowledge process modeling languages (KMLs) follow. A knowledge modeling-specific facet is identified subsequently.

Knowledge Management Concepts: Most KMC approaches consider KM activities, a KM proceeding, and conditions for the realization of KM measures. The consideration of neuronal activities, neuronal proceedings, and an integration with components from a knowledge management domain have not been identified in any approach (see each individual interim conclusion). Since KMCs were defined to manage knowledge in organizations by humans being supported by KM systems, a KMC which is in harmony with the CoNM is still missing per se. Hence, KMCs have neither been evaluated to be the foundation for the CoNM (e.g. as building blocks on various levels), nor have KMCs been evaluated to classify elements required by a CoNM so far.

In order to determine whether current KMCs can at least partly identify a CoNM foundation, or which criteria are suited to be part of a CoNM criteria catalog, the following presents category-wise criteria:

1. Since the CoNM is intended to be implemented in a holistic and general setting and intends to enable KMCs by the operationalization of knowledge, criteria about the concrete management of knowledge is relevant. This refers to criteria from the *knowledge perspective*.

2. Since the CoNM is intended to be carried out in a holistic and general setting and intends to operationalize KM activities, criteria about the situational embedding in business processes are relevant (business process context perspective, behavioral perspective, and functional perspective).
3. Since the CoNM intends to transport neuronal mechanisms in the organizational context, which interprets algorithmic mechanisms as management concepts, criteria about the biology of learning are relevant (neuronal perspective).
4. Since the CoNM intends to establish an association between common KMC and neuronal processes, the CoNM intends to model knowledge processes, such as presented by the GPO-WM or Potsdam KM Model, as well as (artificial) knowledge processes in neuronal networks. This requires individual framework conditions, management levels, and roles.

Since no approach provides concepts about the use of neuronal techniques and neuronal concepts, which is an essential part of the ANN process domain, none of the provided KM frameworks at least partly identify a CoNM foundation. Further, this implies that the processual knowledge modeling domain has not been combined with AI techniques yet. A research gap becomes evident here which requires the following: First, the construction of a CoNM because anything similar is missing so far; Second, a CoNM KM selection system that extends common criteria and can evaluate contemporary KMCs as CoNM foundation; Third, an analysis of currently available KMCs for the use of CoNM relevant aspects, such as neuronal techniques, specific modeling objects, and common tool criteria.

Knowledge Process Modeling Languages: Most approaches suitable as processual knowledge modeling language are based on individual sets of *modeling items* and *syntax*, each having a unique modeling strength. The consideration of neuronal perspectives, neuronal modeling objects, and an integration with components from a processual knowledge modeling domain have not been identified in any approach (see each individual critical appraisal). Since knowledge modeling languages were defined to carry out the processual anchoring of knowledge which does not refer to models obtained from a CoNM, an appropriate CoNM process modeling language is still missing per se. Hence, knowledge process modeling languages have neither been evaluated as the foundation for the CoNM (for e.g., by providing commonly accepted knowledge modeling items and well-defined syntax) nor have knowledge process modeling languages been evaluated as part of a CoNM so far.

To decide whether current knowledge process modeling languages can at least be partly used as a CoNM foundation or which criteria are suited to be part of a CoNM

evaluation criteria catalog, the following category-wise aspects are to be considered: Since the CoNM is intended to be implemented in a holistic and general setting,

1. Any *modeling item* provided by knowledge process modeling languages is relevant as items focus on aspects that are accepted in the process modeling and KM community. When applying accepted concepts from the process domain in the ANN process domain, these criteria are highly relevant.
2. Any *syntax or convention* provided by knowledge process modeling languages is relevant since syntax and convention focus on aspects which are accepted in the process modeling community. When applying accepted concepts from the process domain in the ANN process domain, these criteria are highly relevant.
3. Any *modeling concept* being considered by knowledge process modeling languages is relevant since modeling concepts focus on aspects which are accepted in the process modeling community. When applying accepted concepts from the process domain in the ANN process domain, these criteria are highly relevant.

Since no knowledge process modeling language provides items or syntax for the use of neuronal techniques and concepts, which is an essential part of the ANN process domain, none of the provided knowledge process modeling languages can serve as CoNM process modeling language. Further, this implies that the process modeling domain has not been combined with AI techniques yet. A research gap becomes visible here which requires the following: First, the construction of a CoNM process modeling language because anything similar is missing so far; Second, a CoNM process modeling language selection system, that extends common criteria and can evaluate current knowledge process modeling languages to be part of a CoNM foundation; Third, an analysis of currently available knowledge process modeling languages for the use of CoNM-relevant aspects, such as neuronal techniques, specific modeling objects, and common tool criteria.

Facit: Neither do KMC and KML evaluation frameworks consider the integration of the process domain and the ANN domain nor knowledge process modeling languages do so: Conceptions do not provide ANN and Deep Learning criteria at all. Furthermore, knowledge process modeling languages do not realize the aforementioned criteria. Thus, a research gap within the process domain becomes visible here. This requires the following three: First, the construction of a CoNM KMC, second, the design and demonstration of a CoNM selection and evaluation framework for KMCs and KMLs and third, the implementation and demonstration of a CoNM process modeling language.

2.3 Process Simulation

While process modeling focuses on the structural and behavioral modeling of processes (section 2.1), and knowledge modeling attempts to incorporate the context of knowledge by the provision of structural knowledge elements as well as the consideration of their dynamic effects (section 2.2), the simulation attempts to apply models constructed so far in a simulation environment so that structural, behavioral, and knowledge elements as well as their dynamic effects can be analyzed. In this context, a potential can be identified in the testing of models in closed setups so that the processual behavior of complex structures, such as ANN, can realistically carry out tasks relevant to real-word scenarios. Ideally, the simulation system is built as a symbiosis of real systems and real processes so that ANN components can function live, e.g. as a regulation system, a decision support system, or a recommendation system. The underlying hypothesis of this contribution is that insights in the working of ANN make them ideal for use in real systems, particularly because of the recognition of processual knowledge entities within neuronal structures. Thus, through the interplay of simulation settings and ANN, insights are made accessible for process optimizations, e.g.

The process simulation environment is characterized by the following: First, basic concepts and definitions are established. Second, different kinds of simulation taxonomies are provided based on a literature search. Here, relevant perspectives are identified that will be the foundation for the creation of the *ANN Process Taxonomy* (section 4.1.1) and function as a standard for simulation modeling languages. Third, prominent notations to represent simulation models and embed process models are examined so that it becomes clear the extent to which process models designed can be realized in simulations. Finally, the domain-specific research need is identified.

2.3.1 Basic Concepts and Definitions

Basic concepts refer to the definition of simulation methods in distinction from alternative methods so that simulations can be identified as a foundation for the imitation of neuronal processes (Section 2.3.1.1). The specification of simulation systems follows, which carry out processes and business processes and shall include neuronal mechanisms (Section 2.3.1.2). Based on the definition of simulation models (Section 2.3.1.3), the progress of simulation model creation, what is called simulation modeling, is characterized (Section 2.3.1.4). This serves for the simulation model creation, which is in harmony with ANN model creation and includes the integration with process and knowledge modeling.

Overall, this stands as a foundation for the following two kinds of investigations. First is the examination of simulation taxonomies as objects of investigation since these can be interpreted as building blocks for simulation modeling concepts and consider the processual realization of business processes in organizations. Second is the examination of simulation modeling languages (as further OoIs) to enable the processual simulation of organizational activities. Further, the concepts provided serve as building blocks for the construction of a CoNM.

2.3.1.1 Methods of Simulations

Socio-technical systems – e.g. process models (Def. 3), business process models (Def. 4), enterprise architectures (Def. 7), and knowledge process models (Def. 14) are analyzable based on the assumption that processes can be represented as a recursive, multilevel, and deterministic system of data processing (Dörner, 1999, p.327). The selection of an appropriate research method is derived according to the insight interest (Thim, 2017, p.137). Wilde and Hess present a collection of research methods (Fig. 2.29) which are classified considering the degree of formalization (qualitative and quantitative) and paradigm which is either behavioristic or constructivistic (Wilde and Hess, 2006).

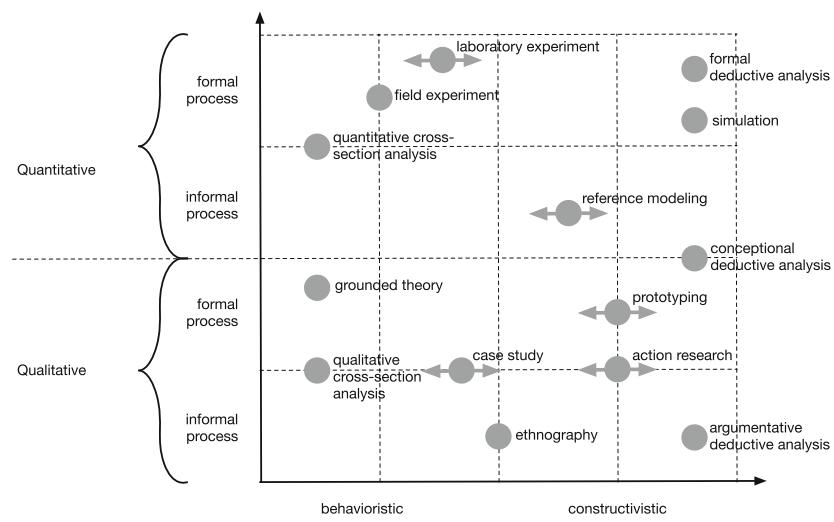


Figure 2.29 Portfolio of business informatics methods (following Wilde and Hess, 2006)

For the CoNM, particular quantitative methods are attractive, which deal with numbers so that a combination with learning methods can be designed. Further, particular constructivistic methods are suitable for the construction of a CoNM since they enable constructive model building, which is similar to the representation or model building of the human brain. The perspective of a system analysis is ideal in this case as first, this allows the systematic examination of a system on the arbitrary level of detail, and second, it allows sub-system building so that a hierarchy of systems is constructed in which systems can be analyzed individually.

Simulation as a Method of System Analysis: A simulation is preferable as a method of system analysis if a problem can be formalized and its analytical solution is (too) complex (Wilde and Hess, 2006). As a computer-based method, it differs from the term “emulation” as it only focuses on relevant attributes and system elements and disregards other elements within the model system (Fig. 2.1). Thus, while the emulation modeling intends to construct *isomorphic* models according to Def. 16, the simulation modeling intends to construct *homomorphic* models (Schütte, 2013, p. 42) which determine the behavior of a system to be examined (Liebl, 1992a, p. 118; Thim, 2017, p. 142). Since relevant parameters are not known initially, the identification of system borders, system elements, element relations, and all three kinds of mappings (forgetful, bidirectional, and extension mappings according to Fig. 2.1) are part of the research progress and are reflected in modeling activities. Fig. 2.30 therefore presents the simulation as research method, which differs from other methods as follows.

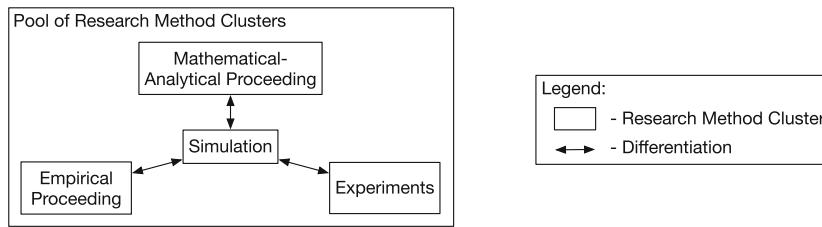


Figure 2.30 Differentiation of the simulation method (following Thim, 2017, p. 140)

Mathematical-analytical approaches are attractive for particularly simple problems to be analyzed (Thim, 2017, p. 136). Approaches abstract the behavior of a system with the aid of premises and assumptions and construct a coherent, mathematical

model. Compared with the original environment (Fig. 2.5), restrictive premises have to be specified during the modeling, so that a rather limited instead of flexible model is constructed. On the other hand, an elaborate and complex mathematical modeling process decreases the transparency of a solution so that the interpretation of analysis results and the transfer of insights to the region of interest is complicated (Thim, 2017, p. 139). Compared to the mathematical-analytical attempts, simulations are more flexible in the functional connection, since they do not require a closed mathematical form and they do not insist on the circumstantial consideration of premises (Thim, 2017, p. 140). Based on the realization of a series of processing steps, simulations realize a more flexible use of decision variables, environment parameters, and decision criteria (Liebl, 1992a, p. 7).

Empirics observe a region of interest and models systems on behalf of these observations. The modeling here is burdened since observations are laborious. Further, not every process to be modeled can be observed in the original environment. Compared to empirics, simulations deal more effectively with the complexity of the original environment (Payne, 1982, p. 273). This is based on the following: First, the processing conditions of simulations can be controlled perfectly (Payne, 1982, p. 3). Empirics would require long-term field studies aiming for the complete survey of relevant parameters by luck or intricate study designs. This includes the observation of different system alternatives which can be observed easily with the aid of simulations. Second, the definition of exact data measure points and the continuous data collection is enabled in simulations (Thim, 2017, p. 139). Thus, the laborious use of measurement equipment and surveying methods can be avoided in empirical settings. Third, the simulation system time can be controlled perfectly. Thus, simulations can effectively slow down the simulation system time and observations can focus on slow effects. Further, the simulation system time can be fixed and effects of long periods can be observed comfortably. Empirics would require the artificial slowdown of empirical settings by circumstantial mechanisms in this case or the conductance of laborious long-term studies (Payne, 1982, p. 2). All this enables the effective observation of feedback effects in simulations.

Experimental approaches intend to establish a controllable laboratory situation to empirically examine a research problem. Considering socio-technical systems of the original environment, the realization of experiments is limited by numerous aspects, such as shortage of resources, laborious design of experiments, complex experiment conditions, and comparisons of system alternatives. Further, the experiment conditions are very difficult to control (Liebl, 1992a, p. 196), which is particularly challenging because of the consideration of human test persons that change their behavior if observed (Shannon, 1975, p. 11). Compared to experiments, simulations enable an isolated laboratory situation which would not be realizable in

experiments. Thus, individual factors are better controllable in simulations than within experiments and simulation-based examinations do not affect the original system. This, for instance, allows the unaffected observation of humans. Further, this allows active management which includes the change of system parameters during the examination and enables the realization of simulation trajectories (Bossel, 2004, p. 18). Since computer-based simulations create virtual environments, the examination of artificial system types, such as ecosystems, is enabled, and simulation-based examinations are much more cost-effective than experiments (Payne, 1982, p. 275).

The *simulation* as a method of system analysis attempts to combine empirical and analytical examination aspects so that disadvantages are neglected (Payne, 1982, p. 273). Generally, it will be applied if a problem can be formalized, if its mathematical-analytical solution is (too) complex (Wilde and Hess, 2006), its empiric examination is too burdensome, and a numerical solution is ideal (Law and Kelton, 1991, p. 1). For this, the behavior of an original system will be expressed as simulation model according to Def. 16, environmental states are expressed by random numbers and analyses are carried out on the basis of this (Shannon, 1975, p. 2). Therefore, mostly, insights are gained by computer-based experiments (Payne, 1982, p. 271).

Simulations and other research methods can be combined to increase insights as follows. First, simulations and empirics can be combined so that simulations consider empirical insights, such as empirical models for the modeling of parameter relations in simulations or the use of empirical data for simulation parameter distributions. Further, simulation results can be the foundation for empirical research. Second, simulations and mathematical-analytical approaches can be combined so that mathematical models can be considered in simulations. Further, simulations results can serve to check assumptions of analytical solutions. Third, simulations and experiments can be combined so that simulations can integrate experiment results, such as experiment data as part of simulations or experiment models for the specification of simulation parameter distributions. Further, simulation results can be the foundation for the design of experiments and validate experimental results.

Kinds of Simulations: The imitation of real-world processes can be realized with the aid of various kinds of simulation methods. While each kind realizes an imitation of real-world processes according to Def. 16, they differ in their technical foundation and are suitable for a certain area of discourse (Fröming, 2009, p. 129). A specific kind of simulation method therefore is to be analyzed with respect to the CoNM construction which will be issued in section 2.3.2.

2.3.1.2 Simulation Systems

As simulations refer to the calculation of mathematical operations, any tool or device which can deal with these operations and replicates operational features of some particular system is referred to as *simulator* (Shishyan and Benndorf, 2017, p.3). According to Mildenberger and Sauerbier (2013, p. 20), simulators ideally consist of the following four separated components: First, *data* represents initial parameters and simulation results. Second, *models* describe the mathematical operations being carried out on data. Third, *algorithms*, operationalize the realization of simulations in regard with different simulation methods. Further, these include storing and visualization functions. Fourth, the *simulation system* integrates data, models, and algorithms to manage their interplay and carry out simulations.

Fig. 2.31 visualizes simulator components and their interdependencies. Particularly, focusing on simulators providing numerous models, classes, and algorithms, the clear component separation is relevant.

In order to stand as universal simulation tool, the following basic functionalities are required (Mildenberger and Sauerbier, 1999; Fröming, 2009):

- User interface and scripting language
- Import of data and models
- Definition and modification of data
- Definition and modification of models
- Definition of initialization parameters
- Modification of simulation parameters
- Report generation
- Analysis tools for simulation results
- Storing and export of simulation results

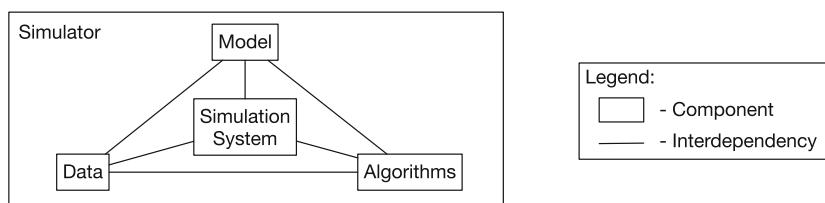


Figure 2.31 Abstract schematic of a simulator (following Mildenberger and Sauerbier, 2013)

As the CoNM is intended to carry out neuronal simulations, a CoNM simulator needs to first consider the separation of the four components introduced and, second, the basic functionalities introduced.

2.3.1.3 Simulation and Process Simulations

A literature search for the two terms “simulation” and “process simulation” yields several descriptions which mostly show process simulations as sub-definition of simulations. Hence, a definition begin with the more general definition of simulations; the latter follows.

Simulations: The term “simulation” can be associated with non-computer-based simulations, e.g. being realized on behalf of a pen and paper. Mostly, the term refers to computer-based simulations. Following the VDI standard, a simulation is a reproduction of a system and its dynamic processes with which one can experiment to gain insights and transfer insights to reality (VDI, 1993).

Some scholars refer to simulations as the imitation of real-world processes or systems over time (Banks, 1998, p. 3) on behalf of mathematical models (Shishvan and Benndorf, 2017, p. 3). E.g. Mildenberger and Sauerbier (1999) consider simulations to be calculations on behalf of mathematical models which means the transfer of an input to an output with the aid of different kinds of simulation methods and their combination. Here, input values can represent the initial state of a system, influence, and disturbance parameters, and output values can represent interim and final states of a system and events occurring during the simulation period (Fröming, 2009, p. 113). Therefore, particular dynamic simulations consider mathematical structures, such as differential and difference equations (Bossel, 2004).

Others refer to simulations as procedure for the realization of experiments with the aid of computers (Piehler and Zschiesche, 2013, p. 5). While using mathematical models, insights about a real system can be gained which are useful for the detection of possibilities for its improvement. Shishvan and Benndorf (2017, p. 3) therefore consider a simulation as imitation of a system’s model operation. Thus, the simulation model is always executable by a computer and its running builds a trajectory of system changes (Grigoryev, 2018, p. 10). Therefore, through the repetitive experimentation and systematic change of system components, the optimal solution is approached by evaluating the model’s output.

Since definitions understand simulations as models, the following defines the term in accordance with Def. 1. It therefore requires the *simulation modeling* activity conducted by a *simulation model creator* for a *simulation model user* with respect to a *simulation purpose* and considered within a *simulation time period* as well as different kinds of sets and mappings visualized in Fig. 2.1.

Definition 16 (Simulation Model).

A simulation model is a model of at least one simulation or a hierarchy of simulations. Each simulation refers to the imitation of real-world processes or systems on behalf of mathematical models which realize the transfer of an input to an output and can be measured by right tools. These simulations, from the perspective of an individual as it is understood, logically form the actual existing world in a multiplicity and become real in the iterative interplay of concrescence in the private and satisfaction in the public.

As the simulation model is designed by a model constructor, the simulation method is considered a construction-oriented method (Wilde and Hess, 2006). While the definition explicitly concerns a “simulation model”, the processual character of the “simulation” as proceeding is supported. The definition recognizes a simulation as detached from a fixed simulation method which is highly dependent on the specific simulation objective (Piehler and Zschiesche, 2013, p. 5). Considering the generality and flexibility of a simulation as methodological strength, the model constructor is challenged with a great adaptation work according to its individual design which is in harmony with the simulation taxonomy (section 2.3.2).

Simulation Process Coupling: The definition about simulation models presented in Def. 16 requires the imitation of real processes which goes beyond the static visualization of process models about dynamic systems (Def. 3). It requires a model-like realization, for instance, on behalf of computers and mathematical models so that dynamic effects of a model system become apparent. Thus, process models can be considered in simulations as simulation objects; this refers to the imitated realization of actual entities in accordance with Def. 3. Alternatively, process models can be considered to model processes with respect to the imitation of real processes. This refers to the interpretation of simulations as OoI and goes along with a process model-based description of the imitation of actual entities or other simulation objects. For example, the following is enabled as follows: the description of scenarios, a hierarchy of simulations, their interactions, and the application of mechanisms.

For the construction of a CoNM, the following therefore distinguishes between the simulation of process models (process models in a passive role), which can be realized on behalf of one collection of techniques and mechanisms and the use of techniques and mechanisms for the simulation design, realization, and analysis. This can result in process models modeled by the simulation (active process models). Having technique-based model creation on behalf of the CoNM in mind, this considers both as model creators: computers and humans.

Neuronal Process Simulation: Considering neuronal mechanisms for simulation purposes requires the application of ANN techniques for passive process models, which refers to the imitative realization of actual entities, as defined in Def. 3, with the aid of neuronal mechanisms. Further, this requires the application of ANN techniques for active process models, which refers to the realization of process models about simulations and results in models of the realization of actual entities, as defined in Def. 3, with the aid of neuronal mechanisms.

2.3.1.4 Simulating

The act of simulating goes along with the following three meanings: First, with a “simulation modeling”, which aligns with the modeling definition (Def. 5) and particularly focuses on the construction of at least one simulation model. Occasionally, a collection of simulation models is required, if several simulation scenarios or variations are intended to be realized (Lass, 2017, p. 164). Second is the “operational simulating”, which focuses on the realization of simulation models constructed and refers to the act of imitating. Third is framing activities drawing the simulation context in the sense of simulation studies. This includes planning and management activities as well as analyses and evaluation of simulation results and focuses on a systematic examination of simulation variations, such as what-if-analyses and sensitivity analyses (Fröming, 2009, p. 114; Thim, 2017, p. 182).

Since the CoNM intends to design simulation models on behalf of ANN learning techniques, the simulation modeling is defined to be the following:

Definition 17 (Simulation Modeling).

Simulation modeling refers to the act of perception of the simulation model creator’s reality, the use of the simulation model creator’s knowledge to apply a simulation modeling language, understanding of the application domain, and realization of the engineering activities in a repetitive, systematic, and principal-based manner such that a simulation model is constructed within a subjective or objective reality.

On top of the simulation modeling, the operational “simulating” considers the act of imitating real-world processes or systems, which is carried out on behalf of simulation models that will be operated by the simulation system (Fig. 2.31). Here, the CoNM intends to consider ANN techniques as well, which differ from ANN techniques for the simulation modeling.

Upstream activities and consecutive tasks going beyond the act of simulation modeling and operational simulating are differentiated further from the perspective of CoNM and connect to the common understanding about simulation. While following a methodological proceeding for simulation studies, the different under-

standings about the term “simulating” depend on the underlying kind of proceeding models. This will be discussed in the following.

2.3.1.4.1 Sequential Models

Most proceeding models refer to sequential models which provide a collection of logically connected activities. As these are carried out during the act of simulating, the term “simulating” is reflected by activities at this kind of proceeding models. Since these mostly provide a linear arrangement of phases which are staged iteratively and are not imperatively required (Law and Kelton, 1991, p. 106), these kind of models are suited to derive tasks for model constructors on an operational level. Hence, the corresponding understanding of the act of simulating refers to a concrete operational level. Fig. 2.32 presents an overview of activities issued by individual models. A clarification is presented in the following.

Initially, nearly all proceeding models provide stages for the *simulation problem definition*: Gilbert and Troitzsch (2005), Doran (1997), Gehring (1998), Law and Kelton (1991), Sterman (2000), Liebl (1992a), Nance (1994), and Gronau (2009). Mostly, this stage starts the model development phase and implicitly involves the act of system boundary specification, which differentiates the model system environment from the original system environment (Fig. 2.1). Liebl (1992a) even explicitly provides a separate stage for *system delimitation*.

As not all simulation models are constructed on behalf of *observation data*, proceeding models distinguish from here on: Some issue the data collection as proper

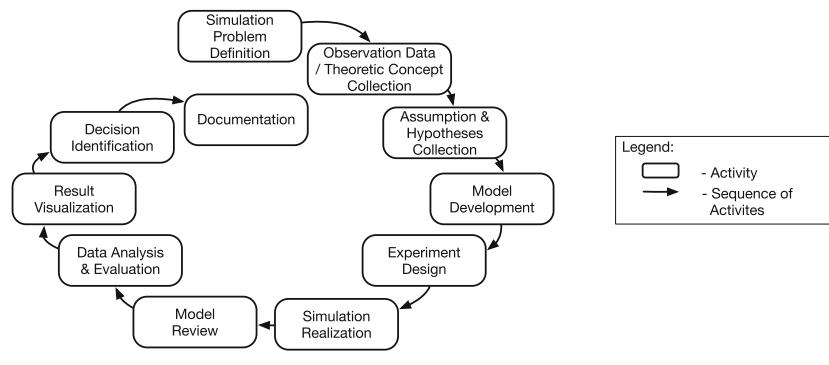


Figure 2.32 Methodology for sequence-oriented simulation studies as overview (synthesized from literature)

phase (Payne, 1982; Doran, 1997; Law and Kelton, 1991; Banks, 1998; Gehring, 1998; Fröming, 2009; Shishvan and Benndorf, 2017), others assume a theory-based model construction (Gilbert and Troitzsch, 2005). Further, at least an abstract phase of system examination is mentioned, which is independent from the choice of data-related empiric methods and theory-based methods (Nance, 1994), or do not concern a data collection at all (Thim, 2017, p. 144). Inspired by empirics, simulation models can consider process models so that they entail process-, activity-, person-, knowledge- and information-related input and output parameters (Gronau, 2009, p. 99).

The proceeding models further distinguish in the explicit consideration of *assumptions*. Some scholars describe assumptions by an individual phase for the establishment of hypotheses (Payne, 1982; Sterman, 2000) and their testing after the simulation realization. Others describe assumptions by the identification of system elements and their interdependencies (Doran, 1997).

Some scholars provide detailed stages about a *model development* which includes the separation of a conceptual, formalized, and implemented simulation model (Law and Kelton, 1991; Gilbert and Troitzsch, 2005; Nance, 1994; Gehring, 1998; Fröming, 2009). This can also include the modeling of processes (Gronau, 2009, p. 99). Others do not provide detailed stages, that are oriented to a differentiation of different kinds of simulation models.

Particularly, for *validation and verification* tasks, a level-based model differentiation is essential since each stage of simulation model can be reviewed separately. For instance, Doran (1997) provides repetitive stages for verification, validation, and sensitivity analyses and Nance (1994) as well as Law and Kelton (1991) stage corresponding review tasks explicitly within the phase of model development so that concrete returns to previous stages are enabled. Gehring (1998, p. 15–38) here only mentions one abstract stage for both the verification and validation of models.

How the reviewed simulation model to be used within the greater simulation context is determined only by some representatives during separate stages for the *experiment design* (Law and Kelton, 1991; Sterman, 2000; Bossel, 2004; Nance, 1994; Liebl, 1992a; Kramer and Neculau, 1998). In this case, the experimentation plan is determined which includes the specification of input parameter, environmental factors and their changes, as well as the number of simulation runs required. Framing activities, for instance, are not considered at all by Hughes (1997), Gehring (1998), Kramer and Neculau (1998), Mildenberger and Sauerbier (1999), Gronau (2009), and Lass (2017).

Proceeding models then jointly then present stages for the *simulation realization* which refers to operational simulation and performs the act of “imitating”. This is followed by *data analysis* stages which examine the performance and stability

of a concrete simulation model in the exploitative or exploratory manner with the aid of KPIs, such as costs, durations, failure rates, and comparison of different simulation runs. This is either realized with the aid of statistical tools or on basis of visualizations. Since the concrete choice of an analysis method here depends on the concrete research objective and research focus, proceeding models do not provide concrete descriptions (Thim, 2017, p. 146).

A target-oriented simulation which prepares the identification of decisions is only proposed by some scholars: Nance (1994) as well as Thim (2017, p. 147). For example, Nance's model considers activities from the very beginning which refers to the problem identification up to the final stage called *identification of decisions*; this includes the integrative consideration of simulation result insights within the original environment (Nance, 1994, p. 1–45). Gehring's proceeding model considers a decision identification implicitly through the stage called *realization of simulation results* and thus includes activities referring to the realization of decisions within the original environment (Gehring, 1998, p. 15–38).

Only some scholars describe the adequate *visualization* of simulation runs (Gronau, 2009, p. 99) and the *documentation* of simulation study results (Nance, 1994; Banks, 1998; Shishvan and Benndorf, 2017, p. 4; Thim, 2017, p. 146), although this is an essential step for the efficient reuse of research insights.

2.3.1.4.2 Level-Based Models

Only some proceeding models consider the interplay of different parties for the methodological founded act of simulating. E.g. Mildenberger and Sauerbier (1999, p.15) and Fröming (2009, p. 122) describe a simulation task processing by three different kinds of departments because of *roles* and *qualifications*.

The *simulation research interested party*, such as a specialized departments having a technical problem or managers having a management problem, present the research need and verbalize a simulation problem. Therefore, a *system analyst* is faced with the challenge of constructing a formal model and considering qualitative and quantitative aspects. The formalized model is implemented by a *computer scientist* who then conducts the simulation. Simulation results are then verified and analyzed by the system analyst. Further, results are transferred from the simulation environment to the original environment. Interpretations and simulation results are presented to the simulation research interest to satisfy the initial research.

In this kind of proceeding models, the act of “simulating” refers to the realization of the corresponding level which includes the realization of the level-based interplay.

2.3.1.4.3 Cyclic Models

Compared with sequential models that mainly consider linearly connected stages, cyclic models establish feedback loops and repetitive stage iterations so that insights of later stages or phases are used to renew previous stages. For example, Nance's model considers a closed stage cycle comprising all stages of the model development phase (Nance, 1994, p. 1–45). The proceeding model is visualized in Fig. 2.33.

Banks (1998) as well as Shishyan and Benndorf (2017, p. 4) establish an interwoven stage cycle: An inner cycle verifies the problem translation from the conceptual model to the implemented model and an outer cycle validates simulation results with both the conceptual model and the data collection. Kramer and Neculau (1998, p. 46) consider two cycles, both starting from the stage for the selection of research methods. Here, one leads to the stage for the parameter identification and the other leads to stage for the design of experiments.

In this kind of proceeding models, the act of “simulating” refers to the realization of corresponding non-stable stages.

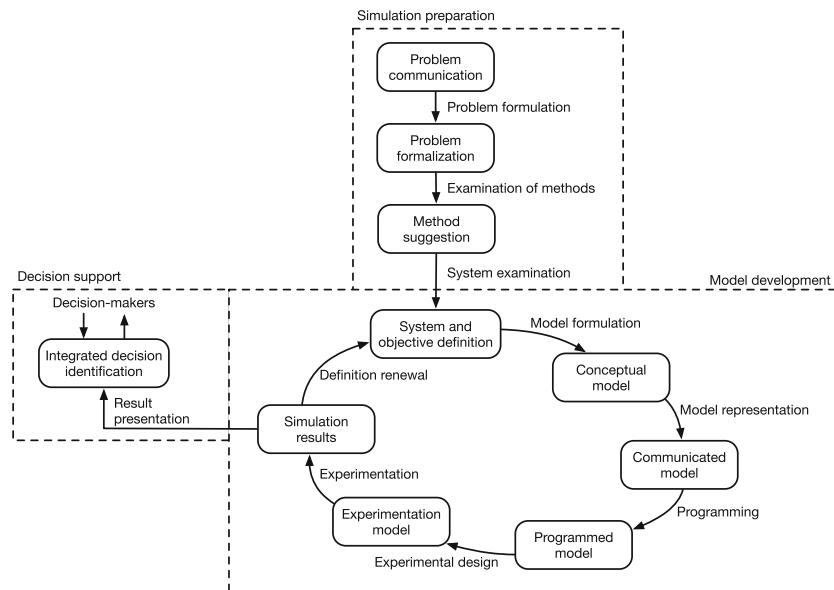
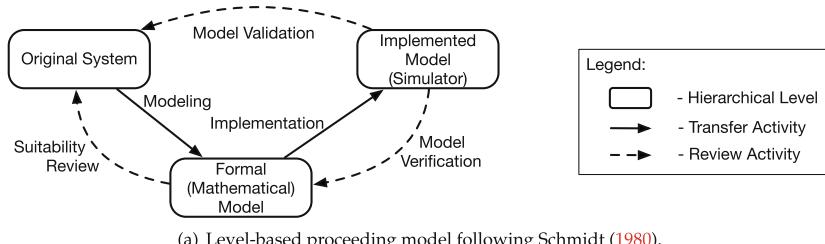


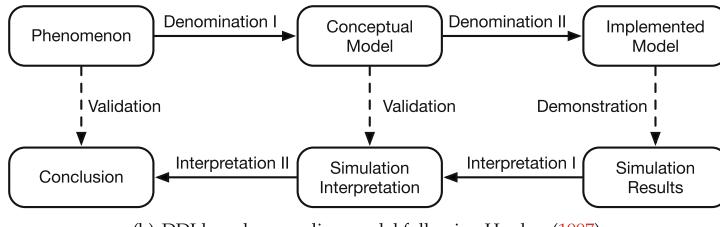
Figure 2.33 Methodology for simulation studies following Nance (1994)

2.3.1.4.4 Hierarchy-Based Models

Further, proceeding models focus on a simulation on behalf of interwoven levels so that a hierarchy of levels is established and activities can be derived from levels (Schmidt, 1980, p. 23; Shishvan and Benndorf, 2017, p. 4; Thim, 2017, p. 143). For instance, Schmidt established a level-based proceeding model focusing on the three review activities of verification, validation, and suitability analysis. It has three levels: First, the original environment, second the formal system model, and third, the implemented model (Schmidt, 1980, p. 23). Hence, the first activity refers to the act of modeling so that a formal model is built on behalf of the original environment. The corresponding review activity refers to the revision about the suitability of the model to imitate the original system (Fig. 2.34 a). A second activity refers to the act of model implementation so that the simulator is set up according to Fig. 2.31. Here, the implemented model has to be *verified* and *validated*. While the first concerns the determination of whether the conceptual model has been correctly translated into a computer program, the second focuses on the determination of whether a simulation model is an accurate representation of the system (Shishvan and Benndorf, 2017, p. 4).



(a) Level-based proceeding model following Schmidt (1980).



(b) DDI-based proceeding model following Hughes (1997).

Figure 2.34 Methodology for hierarchy-based simulation studies

Additional levels are considered by the proceeding model of Hughes, who establishes a focus on the denotation, demonstration, and interpretation (DDI) (Hughes, 1997, p. 334). Since the implemented model is demonstrated, the level called *simulation results* is developed (Fig. 2.34 b). Since simulation results are then interpreted within the conceptual context, the stage called *simulation interpretation* is developed. The obtained interpretations then have to be interpreted within the original context, so that the stage *conclusion* is accessed. Validation activities associate the stages of a *conceptual model* and *simulation interpretation* as well as *phenomenon* of the original system and *conclusion*. Since the implemented model and the original environment are not validated, and verification activities are not provided and a suitability is not issued, review activities of Schmidt's and Hugh's proceeding models establish a clearly distinguishable focus.

In this kind of proceeding models, the act of “simulating” refers to the model transfer from level to level and includes the management of level-based stages.

2.3.1.4.5 Prediction-Oriented Models

If one considers the simulation of a process on behalf of its process steps, one aims to carry out a forecasting (Mildenberger and Sauerbier, 1999, p. 9). Simulation proceeding models particularly focusing on the establishment of predictions and obtain the characteristic to provide simulation systems, that are therefore interwoven with the original system. While simulations are commonly sufficient unto themselves and they are decoupled from the original environment, prediction-oriented simulations require data from the original environment so that a simulation refers to the prediction and characterizes the future original system. Therefore, some proceeding models enable a specialized focus on dealing with prediction-oriented simulations.

Works discussing the same are as follows: Mildenberger and Sauerbier (1999, p. 9), Law and Kelton (2000), Fröming (2009, p. 118), and Shishvan and Benndorf (2017, p. 5). E.g. Mildenberger and Sauerbier's proceeding model focuses on the derivation of predictions for an original system, which is visualized in Fig. 2.35.

While a model is constructed from the current original system at time step t_0 , its simulation output serves as a description of time step t_1 . The output interpretation then serves for the derivation of the future original state which is called prediction. Further, the output results of the simulation model are compared with the output of the original system of time step t_1 , so that an evaluation can be established on behalf of quantitative measures (Law and Kelton, 2000; Shishvan and Benndorf, 2017, p. 5). According to Shishvan and Benndorf (2017), one can e.g. establish the following three: First, a bias, which refers to the sum of differences between all predicted and actual Key Performance Indicator (KPI) values over all n predicted

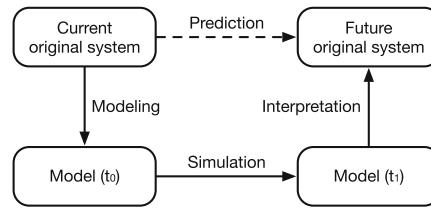


Figure 2.35 Methodology for prediction-oriented simulation studies following Mildenberger and Sauerbier (1999)

time intervals. It therefore represents a tendency of a measurement process to over- or under-estimate the value of a population parameter:

$$BIAS = \sum_n (KPI_{simulated} - KPI_{actual}). \quad (2.9)$$

Second, an average deviation, which is one of several indices of the prediction error and can be defined as the mean absolute deviation between all predicted and actual KPI-values over all n predicted:

$$AVERAGE\ DEVIATION = \frac{1}{n} \sum_n |KPI_{simulated} - KPI_{actual}|. \quad (2.10)$$

Third, an average relative error, which is the average deviation defined in Eq. 2.10 that is divided by the magnitude of the actual value:

$$AVERAGE\ RELATIVE\ ERROR = \frac{1}{n} \sum_n \frac{|KPI_{simulated} - KPI_{actual}|}{KPI_{actual}}. \quad (2.11)$$

In this kind of proceeding models, the act of “simulating” refers to dealing with a model at one time step so that it becomes clear how the model evolves and what it looks like in the consecutive time step.

2.3.1.4.6 Critical Appraisal

Considering the commonly known proceeding models and individual understandings about the term “simulating”, procedures do not restrict the choice of the under-

lying simulation method (section 2.3.2). Thus, the use of ANN mechanisms is not restricted here.

As no proceeding model considers activities for the design, training, testing, and use of ANN (sequential and cyclic models), and decisions with the aid of ANN are not issued as well, models presented are not prepared for neuronal simulations.

Further, level-based proceeding models do not mention the roles of data scientists or corresponding ML expert qualifications that are required for the ANN model construction. Since this is essential for the successful management of ANN, proceeding models are not adequate for a CoNM.

Since no hierarchy-based proceeding model provides stages for the training and testing of ANN, review activities focusing on generalization issues are not provided further. Hence, corresponding activities need to be complemented for neuronal simulations in the sense of CoNM.

Considering the potential of ANN to carry out predictions, prediction-oriented simulation proceeding models do not specify the method of harmonizing a neuronal learning and use of ANN structures to generate predictions within simulations.

Although neither the selection of tools, nor their management is specified in any proceeding model regarding simulations, dealing with ANN in simulation contexts is even more challenging. An adequate simulation proceeding model is therefore not available yet.

2.3.2 Simulation Taxonomies

A simulation method refers to a framework that is used to map a real-world system to its simulation model (Grigoryev, 2018, p. 13). Similar to “terms and conditions” for simulation modeling, the specific kind of simulation method describes how the mapping is technically realized.

As an abstract and flexible methodological concept applied to highly specific contexts, a unique simulation method is not available (Piehler and Zschiesche, 2013, p. 5). Hence, the realization of a simulation is accompanied by significant adaptation work of the simulation model constructor. In principle, the specific kind of simulation methods can rather be determined by a collection of simulation method classes which clarify the kind of simulation to be realized and the adaptation work required to be carried out.

In order to determine what kind of simulation method is suitable for the construction of a CoNM and which adaptation work is required here, the following presents attempts to classify commonly known kinds of simulation methods, which have been identified on behalf of an SLR of the terms “simulation method” and

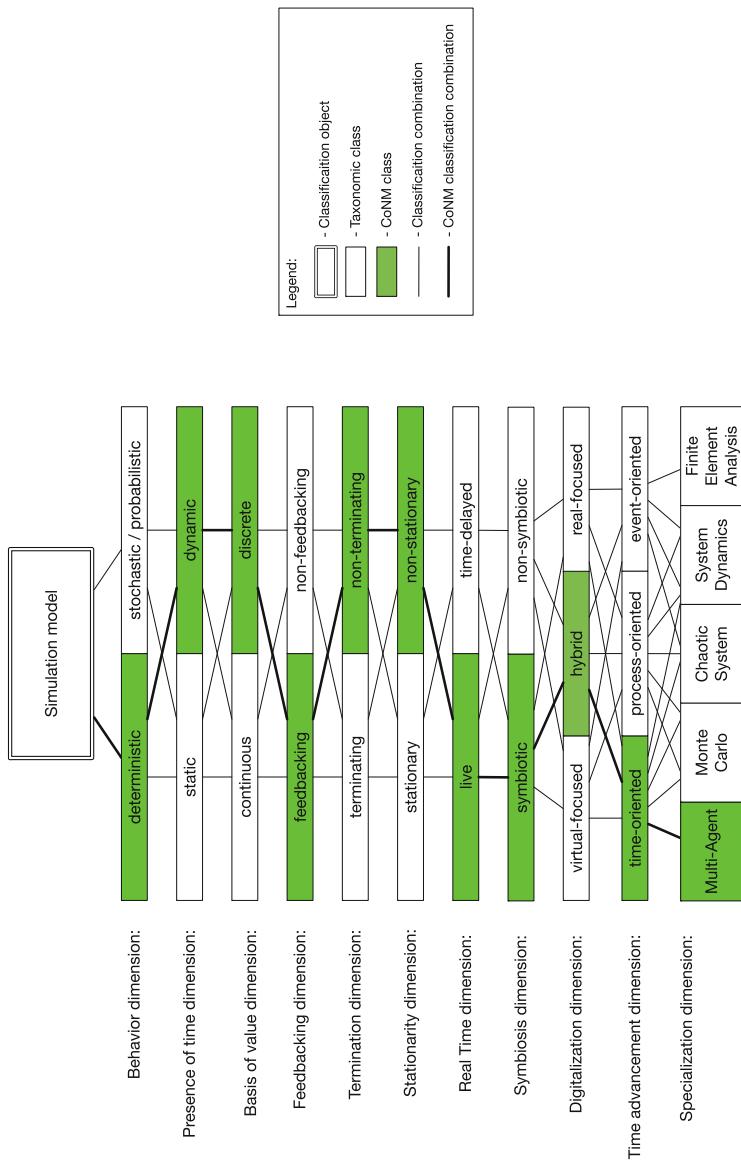


Figure 2.36 Simulation taxonomy and CoNM simulation specification (synthesized from literature)

“simulation classification”. These categorizing attempts are referred to as *simulation taxonomy* from here on, which stands for the classification of simulation models by certain criteria, categories, or classes such that they are integrated in one common system. Further, the classification of CoNM simulations guarantees the conceptual connectivity of CoNM approaches and common simulation methods. The focused taxonomies differ from simulation taxonomies such as design taxonomies describing components and features of simulation tools, architectural taxonomies identifying the type of target systems to be simulated, and usage taxonomies illustrating how tools are used (Sulistio, Yeo, and Buyya, 2004) as the CoNM aims to be realized on a generic, tool-independent level.

In the context of simulation modeling, several taxonomies have been suggested. Here, one can find work by authors such as Mildenberger and Sauerbier (1999), Law and Kelton (2000), Fishman (2001), Sulistio, Yeo, and Buyya (2004), Fröming (2009), Gronau (2009), Babulak and Wang (2010), Piehler and Zschiesche (2013), Küll and Stähly (2013), and Shishvan and Benndorf (2017). In principle, taxonomies comprise a combination of different kinds of taxonomic dimensions and their manifestations, what is called taxonomic classes, which are visualized in Fig. 2.36. Although the dimensions presented are not exhaustive and simulation approaches can be realized without considering all dimensions for a specification, the following strives for completeness so that a CoNM foundation is prepared best.

Specific kinds of simulation methods, which are preferable for the construction of a CoNM need to be selected depending on meta-levels of process models, such as the availability of neuronal perspectives and neuronal modeling objects, and further, they need to be chosen depending on the use of neuronal techniques. The following therefore regards taxonomic classes in separate sub-sections individually and appraises simulation approaches critically regarding their suitability as foundation for the construction of a CoNM. Attractive manifestations have been highlighted in Fig. 2.36 by green and corresponding explanations are concluded in each sub-section.

2.3.2.1 Behavior-oriented Taxonomies

Behavior-oriented taxonomies categorize simulation methods on behalf of the capability to derive the exact simulation result, which is represented by the *determinability* dimension (Sulistio, Yeo, and Buyya, 2004). Its manifestations refer to *deterministic* and *stochastic* approaches, (Mildenberger and Sauerbier, 1999, p. 24; Fishman, 2001; Fröming, 2009, p. 126; Babulak and Wang, 2010; Küll and Stähly, 2013, p. 4; Lass, 2017, p. 70; Shishvan and Benndorf, 2017). Some representatives denominate stochastic approaches as *probabilistic* (Sulistio, Yeo, and Buyya, 2004).

Deterministic approaches enable the imitation of real-world systems having a deterministic behavior: Here, the simulation modeling focuses on a sequence of states, the transfer from state to state, the environmental conditions under which transfers are realized (Babulak and Wang, 2010). Since this is determined perfectly, simulation results only depend on initial parameters and environmental circumstances. So, multiple runs of the same simulation setting result in one and the same result.

Stochastic simulation approaches enable the imitation of real-world systems by stochastic processes and random numbers that model environmental influences (Mildenberger and Sauerbier, 1999, p. 22). Since a modeling specifies the likelihood of transfer from one state to the other, multiple simulation runs of the same initial parameters and environmental circumstances might lead to different simulation results. The strength of stochastic simulation runs therefore lies on analyses about the statistical variance of results (Fröming, Gronau, and Sultanow, 2008) and the security of a statistical statement. The greater the number of simulation runs, the greater the validity of statistical statements.

Interim Conclusion: Since a broad distribution of environmental circumstances will be reflected in trainings and test data for ANN, and the learning of ANN approximates on the basis of variances reflected in data, stochastic relations are considered implicitly by corresponding data sets. The activation of trained ANN results in deterministic behavior, which makes deterministic simulation methods preferable because of the easy integration ability of ANN mechanisms in simulation environments.

Although the use of ANN in simulations does not exclude stochastic simulation methods such as its activation by random numbers, the initial version of the CoNM focuses on deterministic simulations. This reduces complexity because stochastic influences and numerous simulation runs have been considered during analyses for example. Later versions will consider the CoNM in stochastic simulations, of course, so that a greater conceptual connectivity is realized.

2.3.2.2 Presence of Time-oriented Taxonomies

Presence of time-oriented taxonomies categorize simulation methods with regard to the consideration of the time factor within the simulation, which is represented by the *presence of time* dimension (Sulistio, Yeo, and Buyya, 2004). Its manifestations refer to *static* and *dynamic* approaches (Mildenberger and Sauerbier, 1999, p. 24; Fishman, 2001; Fröming, 2009, p. 126; Babulak and Wang, 2010; Küll and Stähly, 2013, p. 4; Lass, 2017, p. 70; Shishvan and Benndorf, 2017).

Static simulation methods enable the imitation of a real-world system at a certain time. Thus, the simulation of one and the same time step always leads to the same simulation result. Hence, the result is independent of the simulation history.

While *dynamic* simulation methods enable the imitation of real-world systems whose states, elements, and processes may vary, the result of a simulation is dependent on the course of simulation. Thus, the simulation of one and the same simulation setting might lead to different simulation results as a different history occurred.

The understanding of conditions for dynamics differs in literature as follows. Bossel characterizes simulation methods to be dynamic if two conditions are fulfilled: First , a system's state changes within a period of time or at least the system can change (Bossel, 2004). This concerns the fact that both need to be examined, the behavior of a system in simulations as well as the capability of a system to be dynamic. While the dynamic behavior of a system can be seen at the varying simulation result, the capability to be dynamic cannot necessarily be seen in the simulation result. Although a system does not exhibit dynamic behavior, it might be capable of the same and would lead to dynamic behaviors in other simulation settings. Second, the system attributes and system element relations change. This implies that systems transform during the course of simulation. Other scholars relax the conditions presented and characterize simulation methods as dynamic if, first, model variables change during the course of simulation (Hoover and Perry, 1990, p. 7) or, second, if the behavior of a system is simply intended to be examined during the course of a simulation (Mertens, 1982, p. 5).

Interim Conclusion: As the ANN can establish both a static behavior and a time-based behavior by recurrent networks (issued in section 2.5.5.3) and the CoNM intends to be capable of imitating dynamic real-world processes, dynamic kinds of simulation methods are preferable as the foundation of the CoNM.

While Merten's conditions for dynamic simulation methods can simply be harmonized with the intention to observe ANN during the coarse of simulations and Hoover and Perry's conditions for dynamic simulation methods can be harmonized with the functioning of ANN based on variables representing neuronal activations that change during the coarse of simulations, Bossel's two conditions for dynamic simulations are challenging as follows.

The use of ANN as simulation components can be aligned with his first condition: if neuronal structures are recurrent then they are capable of establishing dynamic behavior. If their recurrent weights change the ANN behavior during the course of simulation the dynamic behavior even can be observed in simulation results. Further, simply the use of trained ANN in simulations does not satisfy Bossel's second

condition of changed system element relations. Here, the following interpretations can align ANN with the second condition.

As first interpretation, system element relations refer to neuronal weights. Here, a training realized during the course of simulation would satisfy the second condition since weights are changed during learning processes. Alternatively, the modifying influence of neuronal systems on the connection from one neuron to the other would satisfy the second condition of Bossel. When this influence changes during the course of simulation, the system can be identified as dynamic.

As second interpretation, neuronal sub-systems can be interpreted to represent element relations. The varying behavior of these sub-systems would lead to changing element relations of the corresponding neuronal system. Thus, this condition can be satisfied and the simulation system needs to be identified as dynamic.

2.3.2.3 Basis of Value-oriented Taxonomies

Value-oriented taxonomies categorize simulation methods through the capability to simulate infinitesimal small simulation time steps which is represented by a *continuity* dimension (Sulistio, Yeo, and Buyya, 2004). Its manifestations refer to *continuous* and *discrete* approaches (Mildenberger and Sauerbier, 1999, p. 21; Law and Kelton, 2000; Fishman, 2001; Fröming, 2009, p. 124; Babulak and Wang, 2010; Küll and Stähly, 2013, p. 4; Lass, 2017, p. 70; Shishvan and Benndorf, 2017).

Continuous simulation methods enable simulations in which state changes occur continuously during the course of time (Liebl, 1992b, p. 9; Mildenberger and Sauerbier, 1999, p. 21). Here, its variables within one interval represents infinite number of values (Kalisch, 2003, p. 9; Banks et al., 2005, p. 11) so that an arbitrary granular simulation level can be realized.

Discrete simulation methods enable simulations in which state changes occur on a discrete basis at a certain time (Fröming, 2009, p. 124). Here, its variables within one interval can represent a finite number of (Kalisch, 2003, p. 9; Banks et al., 2005, p. 11) so that the simulation granularity can refer to the level of available values only. Even if continuous variables are ready to be considered in simulation runs, their consideration by unsteady simulation processes will implement unsteady state changes only, resulting in discrete simulation methods (Mildenberger and Sauerbier, 1999, p. 21).

Interim Conclusion: Considering ANN mechanisms which operate on behalf of numeric values activation-wise, only discrete simulation methods are suitable for the design of a CoNM.

2.3.2.4 Feedbacking-oriented Taxonomies

Feedbacking-oriented taxonomies categorize simulation methods based on the availability of feedback structures in simulation systems, which is represented by the *feedbacking* dimension. Its manifestations refer to *feedbacking* and *non-feedbacking* approaches (Mildenberger and Sauerbier, 1999, p. 24; Fröming, 2009, p. 126).

Non-feedbacking simulation systems require simulation methods which realize transfers with only one direction of action. In this case, real-world imitations refer to a sequential processing of simple simulation steps.

On the other hand, *feedbacking* simulation systems require simulation methods which realize transfers having more than one direction of action. Thus, the subsequent outputs influence earlier input values and interim results. In this case, different system parts are feedbacking each other. Particular complex simulation processes, such as mental processes, use experiences during the process realization, and the generation of knowledge in a certain simulation setting consider feedback structures among system parts.

Interim Conclusion: The widely used feedback cycles of ANN structures are discussed in section 2.5.5.3; according to these, feedbacking simulation methods are suitable as foundation for the construction of the CoNM.

2.3.2.5 Termination-oriented Taxonomies

Termination-oriented taxonomies categorize simulation methods on behalf of their durability which is represented by the *termination* dimension. Their manifestations refers to *terminating* and *non-terminating* approaches (Mildenberger and Sauerbier, 1999, p. 26; Fröming, 2009, p. 127).

While *terminating* simulation methods realize imitations of real-world processes within a fix period of time, they have a start time and do not stand longer than a certain period of time. Hence, either simulations end after their corresponding life span has exceeded, or they are repeated if certain conditions indicate their start.

Non-terminating simulation methods are not limited by start and end parameters. Hence, they can be used in permanent simulations.

Interim Conclusion: As ANNs are trained on behalf of sequential data having a certain sequence length and recurrent activations are eliminated after each sequence-based activation, these neuronal models can model a terminated behavior. In principle, they can also be considered repetitively in non-terminating simulations runs.

Further, ANN can be trained without eliminating activation histories so that they can stand as non-terminating systems. Although the training of terminating ANN was easier than non-terminating ANN as the correct termination or reinitialization

of ANN was superfluous to be learned, non-terminating simulation methods are suitable as foundation for the construction of a CoNM so that a processual framework can be established that can operate in a non-terminating, permanent manner.

2.3.2.6 Stationarity-oriented Taxonomies

Stationarity-oriented taxonomies categorize simulation methods on behalf of the consideration of a transient response which is represented by the *stationarity* dimension. Its manifestations refer to *stationary* and *non-stationary* approaches (Mildenberger and Sauerbier, 1999, p.27; Fröming, 2009, p. 128).

Stationary simulation methods enable real-world imitations which oscillate before they arrive at stationary states. Here, input values are constant and system parameters are stable (Banks, 1998, p.3–30). Hoover and Perry define the steady state as the state in which the probability to access a certain state does not change any more (Hoover and Perry, 1990, p. 309).

In contrast, *non-stationary* simulation methods focus on the modeling of systems independent from issues of oscillations. From a theoretical perspective, these kinds of systems never arrive at stationary states. Reasons for this can lie in the varying input values, changing system parameters, events causing a structural damage, abrasion (Fröming, 2009, p. 128).

Interim Conclusion: Considering inhibitory and excitatory currents cycling among neurons, the human brain tends to be a non-stationary system. Although the presence of stationary sub-components must be examined, the CoNM focuses on non-stationary simulation systems for a proof of concept.

2.3.2.7 Real-Time-oriented Taxonomies

Real Time-oriented taxonomies categorize simulation methods on behalf of their capability to generate simulation results which is represented by the *real time* dimension. Its manifestations refer to *live* and *time-delayed* approaches (Mildenberger and Sauerbier, 1999, p.21; Fröming, 2009, p. 124).

Live-capable simulation environments enable the interactive management of the socio-economic and environmental system and the simulator (Fig. 2.31) as simulations results are provided almost instantly and can be considered within the original system environment (Fig. 2.1). Particularly in interactive contexts with humans, such as education scenarios or controlling scenes, the possibility to interact with the simulation system is essential since a learning feedback can be generated by the simulation system. Essentially, the capability to present interactive simulation results in real time at a high level of detail is burdened by the simulation calculation effort required and computing power available.

Time-delayed simulation methods enable the imitation of real-world processes, which is decoupled from live simulation settings. Here, data from a live environment can be considered by the simulator, but the simulation results and feedback data from the simulator cannot be considered within the original environment immediately because of its delayed processing.

Interim Conclusion: As the CoNM intends to reflect actual entities and is challenged by huge data streams, a live simulation is hard to realize. Nevertheless, encouraged by an increasing computing power, the CoNM aims to work in live simulation settings. This is particularly essential as realities of several model constructors and model users shall be aligned so that an interaction is enabled as it will be needed for a bidirectional AR process modeling (Grum and Gronau, 2018b).

2.3.2.8 Symbiosis-oriented Taxonomies

Symbiosis-oriented taxonomies categorize simulation methods on behalf of the presence of real-world systems within the simulation setting which is represented by the *symbiosis* dimension. Its manifestations refer to *symbiotic* and *non-symbiotic* approaches (Aydt et al., 2009a; Aydt et al., 2009b; Tjahjono and Jiang, 2015, p. 823).

Symbiotic simulation methods enable simulations that allow a close coupling between the simulation system and a physical system which is usually beneficial to at least one of them. Thus, interactions with simulation components and physical systems are realizable (Fanchao, Turner, and Aydt, 2009). Considering the purpose of a symbiosis, one can find various kinds of symbiotic simulations which refer to systems for controlling, decision support, forecasting, model validation, and anomaly detection as well as their combination which is referred to as *hybrid symbiotic simulation* (Aydt et al., 2009a; Tjahjono and Jiang, 2015).

In contrast, *non-symbiotic* simulation methods refer to simulations which do not consider symbiotic system associations at all. Therefore, the simulation setting only provides model system environments according to Fig. 2.1.

Interim Conclusion: Since the CoNM intends to consider modeling objects inspired from EA (section 2.1.2.3) whose processual appearances (virtually or physically) are within the original system environment and go along with a digital twin being able to be visualized in simulations, the integration of real-systems in CoNM is aimed. Thus, symbiotic simulation methods will be considered as the foundation for the construction of a CoNM.

2.3.2.9 Digitalization-oriented Taxonomies

Digitalization-oriented taxonomies categorize simulation methods on behalf of the consideration of virtual and hardware simulation components which is represented by the *digitalization* dimension. Its manifestations refer to *virtual*, *hybrid*, and *real*-oriented approaches (Lass, 2017, p. 140).

Virtual-focused simulation methods enable simulations which are realized only with the aid of virtual simulation components. This includes e.g. software systems, scripted workers, algorithmic processes, etc. and supports a flexible and cost-effective simulation realization.

Real-focused simulation methods enable simulations which are realized on behalf of real simulation components. Since these include test persons, workpieces, machines, etc. these support the immersion, so that the user's awareness of being exposed to illusory stimuli fade into the background to such an extent that the simulated environment is perceived as real. Thus, these improve communication about demonstration objects, and extend the simulation data collection because of additional sensors equipped in the simulation hardware.

Hybrid-focused simulation methods enable simulations which combine virtual and real simulation components so that advantages of virtual-focused and real-focused simulation methods are realized. This, for example, can be supported as, first, *demonstrators* are built that involve physical attributes such as cubes representing machines or workpieces, and second, these are surrounded by displays that enable the visualization of virtual content (Lass, 2017, p. 141).

Interim Conclusion: As a bidirectional modeling is intended with the aid of AR techniques (Grum and Gronau, 2018b) which enables the capability of hybrid simulation components by further augmentations, including software and hardware augmentations, the CoNM aims to realize hybrid simulations which increases the immersion on behalf of augmented objects. Thus, hybrid simulation methods will be considered as foundation for CoNM simulations.

2.3.2.10 Time Advancement-oriented Taxonomies

Time Advancement-oriented taxonomies categorize simulation methods on behalf of the handling of simulation time which is represented by the *time advancement* dimension. Its manifestations refer to *time-oriented*, *process-oriented*, and *event-oriented* approaches (Mildenberger and Sauerbier, 1999, p. 31; Fröming, 2009, p. 135).

Time-oriented simulation methods realize simulations through a loop which increases the simulation time with every iteration. Calculations within one iteration execute processes of the corresponding time step. When appropriate, the simula-

tion can be terminated if a certain number of loops has been performed or other termination conditions have been achieved.

Process-oriented simulation methods resemble time-oriented simulation methods but they perform simulations through method calls. Here, simulation time can progress between a method call and its realization which is conceptually enabled by the consideration of waiting times (Spaniol and Hoff, 1995, p. 9). While some methods are waiting, other processes can be executed in parallel. Hence, the coordination effort is high although many systems can be modeled efficiently by process-oriented simulation methods (Fröming, 2009, p. 136).

Event-oriented simulation methods realize simulations through *events*. While events are stored in an event queue which is processed on behalf of the event occurrence time, the realization of an event is not simulation time consuming. During the realization of an event, simulation objects are changed which carries out the real-world imitation and includes state changes e.g. The simulation can be terminated either if a certain simulation time is exceeded, a specific event occurs, or the event queue is empty.

Interim Conclusion: Although a neuronal simulation can be realized on behalf of all three kinds of time advancement-oriented simulation methods, e.g. an ANN can be activated as method call (process-orientation) or as a certain event occurs (event-orientation). The CoNM focuses on a time-based activation (time-orientation) since this supports the permanent simulation on behalf of ANN. Ideally, the simulation time increase fits to the ANN activation frequency learned, which is reflected in the ANN's training and test dataset.

2.3.2.11 Specialization-oriented Taxonomies

Specialization-oriented taxonomies categorize simulation methods on behalf of the consideration of specialization-specific simulation concepts, which is represented by the *specialization* dimension. Its primary manifestations refer to *Multi-Agent Methods*, *Monte Carlo Methods*, *Chaotic System Methods*, *System Dynamics* approaches and the *Finite Element Methods* (Mildenberger and Sauerbier, 1999, p. 29; Law and Kelton, 2000; Fröming, 2009, p. 1129; Shishvan and Benndorf, 2017).

Multi-Agent Simulation Methods represent simulation methods which consider multi-agent concepts. Here, active components of a system are considered as agents whose behavior can be specified individually. In a joint simulation, their interactions and interplay are analyzed.

Monte Carlo Methods represent stochastic simulation methods which are realized on behalf of numerous similar random experiments (Mildenberger and Sauerbier, 1999, p. 30; Law and Kelton, 2000; Fröming, 2009, p. 134; Shishvan and Benndorf,

2017). Through their joint examination and with the aid of statistical laws, this includes the use of random numbers and their combination. Analytical problems are solved numerically.

Chaotic System Methods represent simulation methods which deal with chaotic systems (Mildenberger and Sauerbier, 1999, p.29; Fröming, 2009, p.133). Here, even small changes of initial parameters lead to extreme changes of output parameters. Since these can even result from stable systems, a prediction in simulation systems is hardly realizable.

System Dynamics Method is a simulation method developed by Jay W. Forrester. It enables the holistic analysis as well as simulation of complex and dynamic systems (Forrester, 1961; Forrester, 1972; Sterman, 2000; Warren, 2002). The modeling of simulation systems either can be based on a qualitative level which supports the examination of closed chains of effects, or it can be based on a quantitative level with the aid of stocks, flows, internal feedback loops, table functions, and time delays. This supports the examination of repetitive patterns, the so-called *system archetypes* (Forrester, 1972; Senge, 1990b; Senge et al., 1994; Kim and Anderson, 1998; Ossimitz, 2000; Braun, 2002). Since these represent a *reinforcement*, *compensation*, and *time delay*, their identification within a simulation system allows a deeper understanding about the original system and by management, initiated measures can optimize a system effectively. Here, the following three types of diagrams support the proceeding: the Causal Loop Diagram (CLD), the system structure diagram (SSD), and the stock and flow diagram (SFD).

Finite Element Analysis Methods represent simulation methods which focus on the simulation of finite small elements. Intending to simulate spatial distributions and continuous functions, the use of finite small elements is attractive as they serve as approximation of one continuous joint system (Knothe and Wessels, 1999). Therefore, this reduces simulation efforts and enables the simulation of more detailed simulation levels.

Interim Conclusion: Although the list of specialized simulation methods is not exhaustive and there is not only one class of methods preferable for a CoNM, some specialized simulation methods are more attractive than others.

Monte Carlo Methods are not suitable for the construction of a CoNM as the simulation focus does not lie on the numerical solution of analytical problems. Although Monte Carlo simulation settings are not neglected for CoNM extensions, the CoNM focus rather lies on the processual approximation with ANN mechanisms and their neuronal simulation so that an interpretable simulation model and process model is constructed.

System dynamics-oriented methods are not adequate for the construction of a CoNM because the concepts of *reinforcement*, *compensation*, and *time delay* in the ANN domain are attractive but not satisfying with regard to the analysis flexibility. A more granular analysis with the aid of numerical numbers and more powerful algorithmic mechanisms is even more promising than the limited analysis with the three concepts presented by system dynamics.

Chaotic system methods are not suitable for the construction of a first version of a CoNM since its focus shall not lie on the escalation of systems. Although, this is attractive for later versions because of the biological plausible analogy of excitatory escalating neuronal currents, first CoNM versions shall stand as proof of concept for non-chaotic systems.

Finite element analysis methods bring a potential for the construction of a CoNM because various independent simulation components are operating on levels having a different finite element size. Thus, this method specialization is appropriate for the specification and alignment of various levels on one common simulation level. As a proof of concept, a first version of the CoNM assumes to operate on one level initially.

Multi-agent methods are ideal for the construction of a CoNM as various agents can be realized within a joint simulation framework. Here, a flexible interpretation of agents has to be established that allows the use of different ANN techniques, and each is in harmony with neuronal simulations of a CoNM.

2.3.2.12 Critical Appraisal of Taxonomies

Considering the aforementioned taxonomy approaches and its interim conclusions, it seems that today's simulation modeling approaches show numerous aspects, but they are still in development stage and do not consider ANN techniques:

- None of the presented *dimensions* of taxonomies consider neuronal perspectives, neuronal modeling objects, or ANN mechanisms and, therefore, an integration of neuronal simulations (cf. section 2.3.1.3) with the currently available simulation methods is missing.
- Not every *manifestation* of the presented dimensions is suited to characterize a neuronal modeling which is aimed to be carried out with the CoNM.
- Some manifestations are more suited than others to stand as foundation for the construction of a CoNM. An evaluation of currently known simulation methods or concepts useful for neuronal simulations that can be a foundation for the CoNM has not been carried out so far.

A research gap becomes visible here.

A categorization of concepts designed with the CoNM presented here which will stand as enrichment of current simulation methods and techniques will support a broad acceptance of the CoNM in corresponding research communities. Further, a categorization based on a synthesis of taxonomic dimensions ensures its extensive application possibilities. The corresponding broad evaluation of various objects of investigations (OoI) from the simulation domain will therefore support the most effective construction of a CoNM.

Therefore, for the construction of a CoNM, a possibly holistic categorization must be established which still aims for practicability in the evaluation of OoIs. In principle, a practical and holistic synthesis must address the following things:

- It must reflect commonly known taxonomic simulation dimensions and widely accepted simulation manifestations.
- Building on Beer (1979), it must broadly consider the simulation of business processes and reflect their operational behavior. This includes physical flows of materials or components and virtual flows of service (Babulak and Wang, 2010).
- Building on Grum and Gronau (2018a), it must address the knowledge construction and flow of knowledge on a neuronal level. This includes, but is not limited to, abstract knowledge simulations such as knowledge flows of an processual knowledge management (Fröming, 2009) and knowledge workers (Babulak and Wang, 2010).
- Considering the model construction mentioned by Kueng et al., it must describe the process of simulation modeling (Kueng et al., 1996, p. 98). Here, the kind of information required as simulation input for the methodology and output from its application in simulations, such as process related data, (Babulak and Wang, 2010) must become clear.
- Building on Scheer (2002, p. 112), it must not be limited to the design of simulations but must consider tools, methods, and further architectural components which are relevant to carry out neuronal simulations.
- ANN techniques must be harmonized with the understanding of simulation modeling and knowledge utilization.

The creation of a CoNM will therefore consider those points and it will be based on a practical synthesis of dimensions presented here.

2.3.3 Simulation Modeling Languages

Considering basic simulation modeling concepts and definitions (section 2.3.1) as well as taxonomic manifestations and dimension-based standards from simulation taxonomies (section 2.3.2), the following examines prominent notations for the representation of simulation models, the so called *simulation modeling languages* (SMLs).

On the basis of construction-oriented and system theoretic definitions of simulation models (Def. 16) and definitions of simulation model creation (Def. 17), simulation modeling languages are defined as follows:

Definition 18 (Simulation Modeling Language).

A simulation modeling language (synonym for simulation modeling notation) is, not necessarily but mostly, a graphical notation satisfying more or less sophisticated syntactic rules to be applied by a model creator to support the construction of a model creator's reality and create a representation of its subjective or objective reality. This reality concerns imitations of real-world processes and systems, the so-called simulation model within the creator-specific subjective or objective model environment on arbitrary levels of abstractions and granularities of deep structures.

As the definition does not exclude specific simulation models, it considers the specialization of domain-specific simulation models, such as simulations of business process models or process models for ANN. Here, each domain requires individual simulation modeling elements being in coherence with the most abstract and deepest structured meta-model available (see Fig. 2.2). Although this relation is not visible when considering a specific simulation model example, and most simulation modeling notations do not satisfy this formalization adequately because they only address very specialized issues and only a small selection of common modeling perspectives (cf. section 2.3.2), this relation is essential as it holds the (modeling) world together.

According to the *modeling definition* (Def. 5), during the act of simulation modeling, knowledge is considered by the model creators follows: First, knowledge is considered to abstract from a sophisticated reality by the model creator (see *forgetful mapping* of Fig. 2.1). Second, knowledge is considered to complement models with attributes (see *extension mapping* of Fig. 2.1). Third, knowledge is considered by the model creator to efficiently and appropriately express representations on behalf of the corresponding simulation modeling language and its conventions. Since these kinds of knowledge are available within the model constructed in a tacit manner, the subjectivity and the hardly comprehensible view of the model creator leads

to a complicated tension between original creator-model relationship presented in Fig. 2.5.

The term *mostly* in the definition implies that simulations can be carried out by a combination of *graphical modeling items*, but can also refer to *mathematical expressions* or a corresponding representation as *script language* in the sense of computer-aided software engineering (CASE) as well (Case, 1985; Bergin, 1993; Bucci et al., 1994; Lang and Duggan, 2001). The common denominator here is the use of language constructs, such as *words* and *grammatics*. By following transformation rules, the models following different notations or representation forms can be *translated* into one or the other language. As the corresponding two models are well formalized, this translation can be carried out without information loss or information writing, which refers to the same level of *information entropy*.

Selection of representatives: As many simulation modeling notations can be found in the literature, the selection of adequate representatives must be addressed. In principle, process modeling languages are suitable to at least partly create simulation models. Suitable approaches therefore can be found at the Business Application Research Center (BARC) Würzburg and Fraunhofer Institute as well which focus on the analysis of BPMN, BPEL, BPEL4WS, EPK, Entity-Relationship-Models ERM, Ishikawa, IUM, Kommunikationsstrukturanalyse, Line of Visibility Enterprise Modeling LOVEM, PACE, Petri-Nets, Semantic Object Model SOM, Unified Modeling Language UML, and XPDL (Böhn, 2007; Böhn, Burkhardt, and Gantner, 2010; Böhn et al., 2014; Drawehn et al., 2014, p.38). Since these have been described as PMLs in section 2.1.3, the following particularly focuses on simulation modeling languages and the contribution therefore selects simulation modeling languages according to the following criteria.

1. Modeling languages having a broad market distribution which was determined by market research studies (Drawehn et al., 2014).
2. Modeling languages having a broad acceptance in the modeling community
3. Modeling languages being applied in at least one scientific publication

Adequate examination level: In order to satisfy requirements for an adequate management of objects of investigation (OoI) in terms of SLR (Levy and Ellis, 2006) which corresponds to the appliance of Bloom's taxonomy (Bloom et al., 1956). Each modeling language is described by the following categories:

- The short *abstraction* which serves as a kind of introduction to simulation modeling languages on the same, abstract level

- Its individual *description* which characterizes the individual OoI
- *Side facts* issuing the OoI-specific development and historic evolution
- The provision of a *meta-model* visualizes modeling items and its underlying syntax This is completed by *Modeling conventions* clarifying how the meta-model of the OoI is applied so that concrete process models are constructed.
- An *example* visualizes the appliance and underlines the simulation modeling strengths of the OoI.
- Finally, the *critical appraisal* evaluates whether and how the individual modeling language is suited to stand as foundation for the CoNM.

Thus, according to Bloom's Taxonomy, categories considered for any simulation modeling language correspond to the levels of *knowing* and *comprehending* if OoIs are described and side facts are presented, *applying* as a meta-model is generated and an example is presented, *synthesizing* as a common level of understanding is worked out as well as *analyzing* and *evaluating* as common analysis criteria are issued and the simulation modeling languages are regarded with respect to the CoNM.

2.3.3.1 Space Markup Language

The Space Markup Language (SpML) is a graphical modeling language which enables the construction of simulation models showing the physical positioning of simulation objects. Thus, it is suited for the modeling of the resource-oriented and organizational perspective on a system (section 2.1.2.1). As a key modeling approach for simulation models, the SpML is a key element in the software tool known as *AnyLogic* (section 2.6.1.6).

Description: Based on the two assumptions of first agents not interfering with one another and second network segments not being limited by capacity issues, the SpML creates a network of nodes and paths, so that the movement of agents can be specified within the simulated, physical space (Grigoryev, 2018, p. 137). Within this network, *nodes* represent the place where agents might reside or perform an action and *paths* stand for the route agents can be used in order to move between nodes. The SpML element called *attractor* then allows the control of agent locations inside a node. Examples for this are the viewing direction of agents performing an action, the seating position of agents waiting, or the interaction specification of two agents.

As presentation elements, such as lines, text, pictures (e.g. CAD drawings and maps), as well 3D objects can be associated with simulation objects, spatial realistic 2D and 3D animations can be set up and presentation issues, such as the attention guidance, can be realized efficiently.

Since certain SpML elements are conceptually accompanied by simulation logic, specialized functions are incorporated into the simulation model. Examples refer to *pallet racks* having a capacity for the storage of products, *walls* hindering agents to cross a certain area, as well as target and service lines and areas positioning stochastic distributions within the simulation space.

Side facts: First, concepts for a graphical environment where any bit of specification is animated and accessible were developed by the *Distributed Computer Network* research group at Saint Petersburg Polytechnic University (Borshchev, Kar-pov, and Roudakov, 1997) and were named *COVERS* (**C**Oncurrent **V**ERification and **S**imulation). Here, the consideration of physical objects aside from human users and computers in one environment was initiated. As *The AnyLogic Company* was founded by this research group, the company overtook the development. The SpML development was professionalized and a key element in the tool developed. Since then, four major releases have been published and the most current version of the SpML is available in *AnyLogic* of vers. 8.3.3 (The AnyLogic Company, 2019).

Meta-model: Fig. 2.37 presents the general *SpML meta-model*. Here, it becomes clear that SpML elements are related by four different kinds of associations. First, the *agent network association* constructs the network of nodes and paths. Second, the *contains association* brings conceptually accompanied elements in the network specified. E.g. a path is complemented by a pallet rack, or a node is complemented by an attractor. Third, the *processual association* connects stochastic elements by process models. Although any modeling language per se is suited to specify this relation, AnyLogic considers *flowcharts* for this (section 2.3.3.2). Fourth, the *interpretative association* considers interpretations of the model constructor but are not comprehended by the simulation system itself. Examples refer to the node design on behalf of an underlying map, the wall construction on behalf of a spatial sketch or 3D object positioning.

Considering the presented general simulation model, SpML-based simulation models are constructed by the following SpML design rules (Grigoryev, 2018):

1. SpML's core nodes refer to *nodes* and *paths*.
2. Nodes are directly connected with paths. Hence, nodes and paths might not be the predecessor or successor of the same element type. Thus, SpML-based simulation models consist of the alternating elements of *nodes* and *paths*.
3. SpML-based simulation models start and end at a *node* or *target line* so that one can define an area where a process and the simulation is intended to begin and finish.

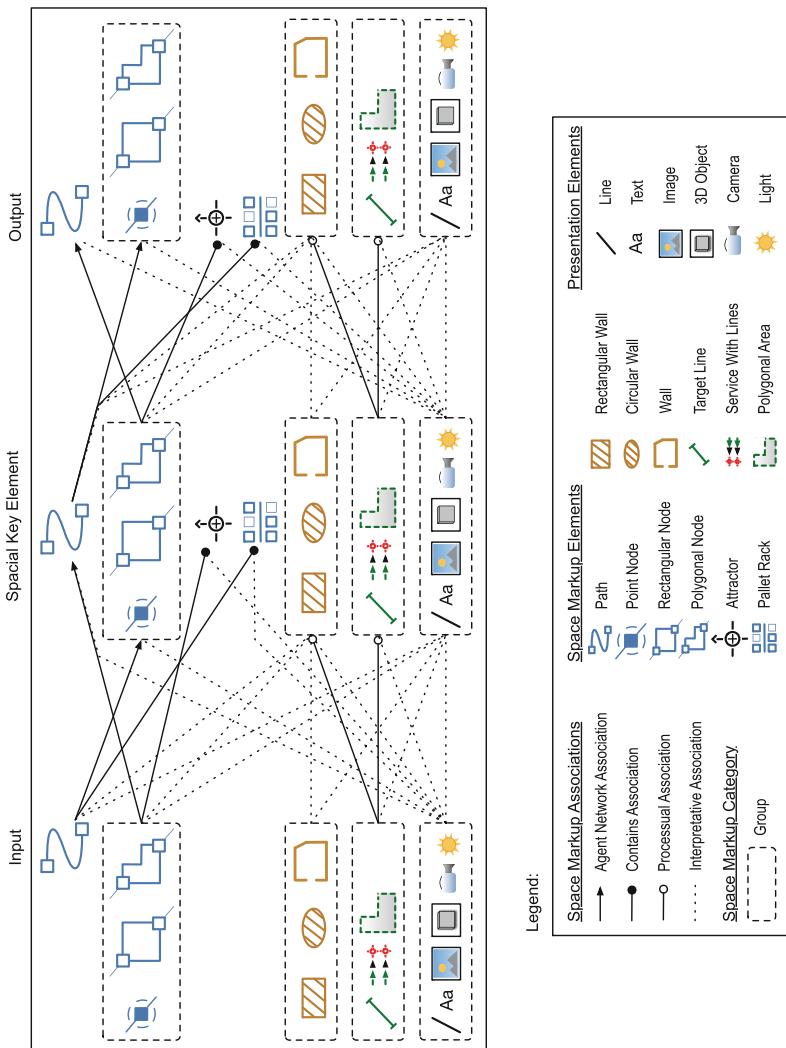


Figure 2.37 Generic Space Markup Language meta-model (synthesized from literature)

4. SpML-based simulation models at least consist of one atomic node or a composition of further nodes.
5. Paths connect nodes corresponding to the presence of ways among nodes. Thus, they are bidirectional per se but can be unidirectional as well.
6. Each path has only one incoming and one outgoing node.
7. SpML-based simulation models position *attractors* within nodes and *pallet racks* along paths.
8. SpML-based simulation models relate stochastic elements by *processes*.

Example: An example model for the SpML is visualized in Fig. 2.81. It shows the simulation of a serpentine queue and thus clarifies SpML conventions.

Within the center of the modeling area, one can see the sketch of the hall to be simulated. Since geometric dimensions are considered here, the spatial realistic modeling is supported. Based on the sketch, *walls* can be set up efficiently (highlighted in yellow). Two *target lines* are positioned within the model, one on the very left and one on the very right of the hall (both highlighted in green). While the left represents the starting point of pedestrians, the right represents the destination. A squared rectangle node (highlighted by a green, dashed line) represents the waiting area. Next to the waiting area, three *service with line* elements can be found. Here, pedestrians from the waiting area have to line up at one of three queues. Then, they are controlled by security experts. Although individual elements do not specify the simulation model entirely, particularly the spatial arrangement can be specified efficiently.

Critical appraisal: Since the SpML is not a process modeling language, the specification of elements required for business processes are merely supported on behalf of laborious code implementations and parameter designs. Therefore, modeling language criteria from the business process context and organization perspective can only hardly be satisfied. Modeling language criteria from the behavior, function information, and knowledge perspective cannot be satisfied at all since modeling on behalf of the SpML focuses on spatial relations only.

The SpML clearly shows strengths in modeling language criteria from the simulation domain and simulation tool criteria. It manages simulation time, the parameter initialization, and event-based changes seriously so that scenarios can be realized efficiently. Although a visualization of different scenarios and a hierarchy of scenarios is not enabled, what-if scenarios on behalf of arbitrary complex KPIs are supported. Further, since 3D animation and control is supported conceptually, an adequate foundation for AR is realized.

Although various kinds of simulation methods are supported by the SpML, ANN modeling objects, concepts, or mechanisms are not issued at all. Thus, modeling language criteria from the NN and ANN domain as well as ANN tool criteria are not met at all.

So, particularly in regard with the spatial modeling, the SpML is suitable as a foundation for the CoNM. Here, modeling entities can serve as building blocks for a CoNM. Further, the conceptual fuse of modeling elements and 3D objects and simulation mechanisms is suitable so that the SpML can complement non-spatial modeling languages of both ANN and business process domain.

2.3.3.2 Flowcharts

The **flowchart** is a graphical modeling language which enables the construction of models about interrelated information in an organized fashion (Harris, 2000, p. 153). This includes e.g. the sequential or chronological visualization of events, process steps, functions, etc. As links of flowchart nodes are proportional to the element they represent, a *proportional flowchart* seems more precise (Harris, 2000, p. 154). Since flowcharts can be used to create a diagrammatic representation of any systems and programs, an algorithm can be visualized similar to a step-by-step approach to solve a task (IBM-Corporation, 1970). Thus, flowcharts are suited to construct simulation models according to Def. 16 and they are suited for the modeling of resource-oriented, behavioral, and functional perspective on a system (section 2.1.2.1). So, flowcharts are often key elements in simulation tools such as one can see at the example of *AnyLogic* (section 2.6.1.6).

While the term *flow diagram* is often used as a synonym for *flowcharts* (Harris, 2000, p. 153–155), the term can be used as counterpart as well (Harris, 1999, p. 156). Thus, the flow diagram rather refers to a graphical representation of the physical route or flow of people, materials, paperworks, vehicles, or communication associated with the flowchart of processes. Thus, they represent a combined visualization of spatially distributed physical objects and process tasks. Fig. 2.38 (a) presents an example for this.

Here, physical objects are visualized by blanc shapes such as *racks*, *counters*, and *cashiers*. Furthermore, gray colored shapes represent different kinds of process tasks. Examples refer to a *transportation*, a *delay*, an *operation*, and an *inspection*. Since the space using process tasks such as the customer's movement are represented by edges connecting the gray shapes, a control flow visualization comparable to common process modeling languages (section 2.1.3) is not visualizable. Hence, a behavioral perspective cannot be visualized on behalf of the flow diagram and thus, alternative routes, decisions, and parallel process task realizations are not visualizable. Visualizations of flow diagrams thus rather refer to models in terms of

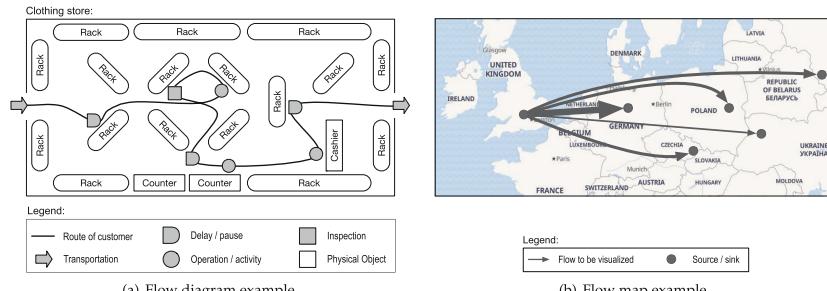


Figure 2.38 Differentiation of flowcharts (created to show the strength of the object of investigation)

historic process instances or currently made decisions implicitly shown by their final process instance. A further difference from common process modeling languages is the missing process task description. As different process tasks can only be visualized by different gray shapes, flow diagrams are not suited for arbitrary complex models having a high number of different tasks and decisions. Since these are focused on by the CoNM, flow diagrams are not considered any further for this point.

As flow diagrams consider the quantification of flows in addition which superimpose floor plans or maps by the thickness of flow visualizations e.g., one speaks form *flow maps* (Phan et al., 2005, p. 156). Examples refer to the number of people in a migration, the number of packets in a computer system network, or the amount of goods being traded (Phan et al., 2005) and Fig. 2.38 (b) presents an example visualization for the same. On behalf of this kind of diagram, the movement of objects from one location to another can be visualized. Thus, edges represent corresponding process tasks so that each process start is spatially or geographically anchored by *sources* and each process target is spatially or geographically anchored by *sinks*. Since the processual task is realized between two coordinates, the process understanding rather refers to processes based on a relational constituted being (section 2.1.1.3) than an understanding which aligns with actual entities of Def. 3. As only the latter supports the construction of the CoNM, this is because the individual processual element is satisfied with a one-point position in time space and with this, the spatial network construction of process tasks is enabled, flow maps are not considered any further here. The following therefore focuses on commonly known flowcharts apart from *flow diagrams* and *flow maps* as these are more preferable as a foundation for the construction of a CoNM.

Description: The modeling of flowcharts is realized considering only one perspective. This visualizes the structure of a process by *operations* which can represent activities, tasks, program steps, functions, etc. (Harris, 1999, p. 156). The *control flow* is visualized by arrows connecting operations and the Boolean operator XOR which is called *decision*. As *documents* take part in the operation realization, the corresponding shape is associated with an *unspecified a flow operation*.

Side facts: First, concepts for flowcharts can be found at Frank and Lillian Gilbreth in 1921 (Gilbreth and Gilbreth, 1921). An initial standard was derived from Gilbreth's original work by the AMSE standard (American Society of Mechanical Engineers, 1947) and a variation was developed to plan computer programs (Hartree, 1949, p. 112) which made the modeling language ideal for information science (Neumann, Taub, and Taub, 1963, p. 80-151; Goldstine, 1972; Bashe et al., 1986, p. 327). In 1970, flowcharts became popular for describing algorithms and since then, certain variations and specializations have been developed, such as more sophisticated activity diagrams (section 2.1.3.3).

Meta-model: Fig. 2.39 depicts the general *flowchart meta-model*. Here, it becomes clear how the basic diagram type is sufficient with the aid of only some modeling entities. Considering an extensive collection of flowchart approaches, the following only focuses on the most common flowchart symbols (Harris, 1999, p. 156).

Considering the presented general flowchart model, flowchart-based simulation models are constructed by the following flowchart design rules. Because of the lack of universally accepted standards and the versatility of flowchart variations, the appearance shapes vary significantly (IBM-Corporation, 1970; Harris, 2000, p. 155), and various conventions have been established. For example, one can find the American National Standards Institute (ANSI) of the 1960s (Shelly and Vermaat,

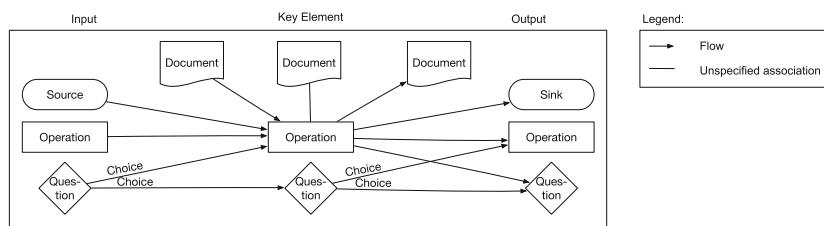


Figure 2.39 Generic flowchart meta-model (synthesized from literature)

2011), the International Organization for Standardization (ISO), that adopted ANSI symbols in 1970 (Myler, 1998), or the more modern ISO 5807 in 1985 (ISO Central Secretary, 1985).

Thus, the following must be interpreted as a common tenets of the aforementioned aspects.

1. In general, flowcharts flow from top to bottom and left to right.
2. Flowcharts' core nodes refer to *sources*, *sinks*, *operations*, *decisions*, and *documents*.
3. Flowchart-based models start with a *source* node and end with a *sink* node so that one can define an area where a process and the simulation is intended to begin and finish.
4. The operation's name should reflect its time-consuming perspective of the accomplished operation.
5. The decision's name should reflect its question to be answered by the decision.
6. The document's name should reflect its meaning for the process.
7. Nodes are directly connected with edges representing a *flow*. Hence, nodes might not be the predecessor or successor of nodes and flowchart-based models consist the alternating elements of *nodes* and *edges*.
8. Each edge has only one incoming and one outgoing node.
9. Operations can be associated with artifacts, such as *documents*, by an edge called *unspecified association*.
10. The decision's outgoing edge name should reflect choices selected by the decision.
11. Flowchart-based models at least consist of one atomic operation or a composition of further operations.
12. Control nodes are represented by rectangles and can be divided into upper and lower parts. Here, basic rules only refer to *XOR* rules.

Example: An example model for a flowchart is visualized in Fig. 2.40. It shows the simulation of three serpentine queues on behalf of the modeling and simulation tool called *AnyLogic* whose context is clarified with Fig. 2.81. The figure presented here provides its corresponding flowchart and thus clarifies flowchart conventions.

At the center of the figure, the flowchart is visualized in green. It refers to the process model of pedestrians walking through the hall from the left to the right. It consists one source on the very left, three consecutive operations, and one sink on the very right. Since geometric dimensions are specified by the *Space Markup Language* model (see section 2.3.3.1), processual elements of the flowchart can be positioned within the simulation space. For instance, the pedestrians appear at the

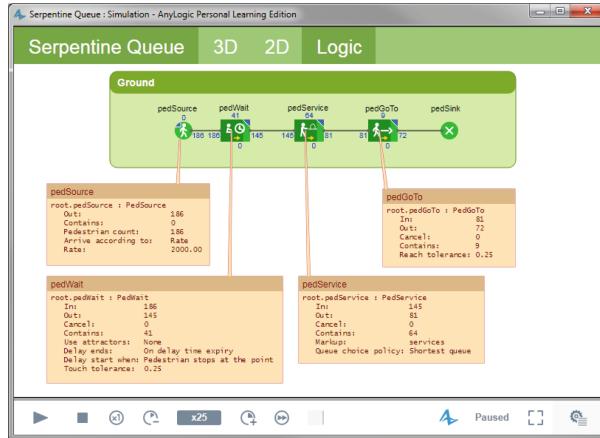


Figure 2.40 Flowchart-based simulation model example (created with the aid of AnyLogic to show the strength of the object of investigation, c.f. section 2.6.1.6)

target line element on the left of the hall (Fig. 2.81), which is specified by the *source* element of the flowchart, and disappear at the *target line* element on the right of the hall (Fig. 2.81), which is specified by the *sink* element of the flowchart. The fact that pedestrians are intended to wait within the waiting area (green rectangle in Fig. 2.81) is specified by the operation element called *pedWait* of the corresponding flowchart. Since pedestrians shall cross one of three security checks, the operation element called *pedService* makes agents walk through one of the three spatial elements called *service with line* of the corresponding Space Markup Language-based model. Then, the operation element called *pedGoTo* specifies that agents shall go to the *target line* element to the very right of the hall to be simulated.

At the bottom of the flowchart of Fig. 2.40, orange blocks provide further information about each flowchart element during the current simulation run. An overview of this is presented aside each flowchart element in blue. Overall, the flowchart model effectively controls agents by processes and the linkage to the SpML model allows the positioning within the simulation space.

Critical appraisal: While referring to non-specialized flowchart modeling languages providing common modeling entities, process visualizations only enable simple sequential processes. As common Boolean operators are not supported circumstantially, the *decision* element is the only element provided for this. The behav-

ioral perspective of a system cannot be modeled in a satisfactory manner. Hence, modeling language criteria regarding a system behavior, its processes, and functions are only satisfied in a rudimentary manner. Although enabling program modeling by definition includes the creation of simulation models, the systematic specification of simulation runs and scenario-based experiment designs is not enabled by flowcharts. It enables only the modeling of simple simulation runs.

A particular focus on business processes as well as knowledge management domain is not established at all. Further, modeling elements from the NN and ANN domain are not considered at all. Hence, the corresponding modeling language criteria are not fulfilled.

As neither simulation nor ANN mechanisms are considered by the modeling language of flowcharts, the preparation of models to be realized by simulators and ANN tools is not realized at all. Here, circumstantial expansions would be needed to make flowcharts a foundation of the CoNM.

2.3.3.3 System Dynamics

System Dynamics (SD) refers to both a simulation method as it was described in section 2.3.2.11 as well as a graphical modeling language which enables the construction of simulation models showing dynamic effects in socio-economic systems. Thus, it is suited for the modeling of the resource-oriented and activity-oriented aspects of a system (section 2.1.2.1).

Description: The simulation modeling in SD is realized with the aid of the following types of diagrams (Sterman, 2000):

The first type of SD diagram refers to the Causal Loop Diagram (CLD). Here, the focus lies on the qualitative identification of closed chain effects. This is modeled by representations about a system's components, here referred to as *variables*, and their interactions which act continuously and are referred to as *causal influences*. Interactions are characterized by both their *causal direction* from cause to effect and their *polarity*. Either they have a positive polarity, so that the effect indicated increases if the cause increases, or negative polarity so that the effect decreases if the cause increases (Sterman, 2000, p. 139). If a closed interaction loop is created having interactions with the same polarity, a *reinforcement* is established. If an interaction loop is established having interactions with different polarity, a *balance* loop is established. Further, interactions can be characterized by the availability of a *delay* so that their effect is applied by a retard. Since the interaction strength and delay time is not specified any further, the CLD is not adequate to represent the behavior of a system (Richardson, 1986). Thus, the CLD is rather suited to start modeling and capture mental models to develop hypotheses about the causes of dynamics

explicitly and to communicate about important feedback circles (Sterman, 2000, p. 137). As the CoNM intends to quantitatively consider processual relations and focuses on the dynamic behavior of a system to be modeled, the CLD is not suitable to be considered any further.

The second type of SD diagram refers to the stock and flow diagram (SFD). Here, the focus lies on the quantitative assessment of chain effects which for e.g. already have been identified by the CLD. Here, *stocks* quantitatively operationalize components as they might have been identified by the CLD. These represent any entity which has been accumulated or depleted over time so that we have

$$Stock(t) = \int_{t_0}^t [Inflow(s) - Outflow(s)]ds + Stock(t_0). \quad (2.12)$$

In this equation, the *Inflow(s)* stands for the value of the inflow at any time s between the integral start, which is the initial simulation time t_0 and the current simulation time t (Sterman, 2000, p. 194). Then, according net rate of change of any stock, its derivative therefore refers to

$$Stock(t)/dt = Inflow(t) - Outflow(t). \quad (2.13)$$

Stocks therefore enable the following: First, stocks enable the characterization of the state of a system so that actions can be determined. Second, they establish systems with inertia and memory since past events can be reflected by accumulations of any stock. Third, they are the source of delays so that they decouple rates of flow and enable disequilibriums (Mass, 1980). *Flows* further specify how stocks quantitatively change and *delays* determine when flows tend to start. On the basis of these simulation modeling entities, it can be analyzed as to how far feedback loops result in a *reinforcement, compensation, oscillation, aging behavior* or repetitive pattern, what is called *system archetypes* (Forrester, 1972; Senge, 1990b; Senge et al., 1994; Kim and Anderson, 1998; Ossimitz, 2000; Braun, 2002; Fröming, 2009, p. 130).

Side facts: SD initially was developed by Jay W. Forrester as a simulation method to enable the holistic analysis of systems as well as the simulation of complex and dynamic systems (Forrester, 1961). Since then, it has gained popular acceptance in the simulation research community (Forrester, 1961; Forrester, 1972; Sterman, 2000; Warren, 2002) and various notations and variations differing only slightly have evolved (Kim, 1992; Senge et al., 1994; Richardson, 1997; Sterman, 2000, p. 140).

Meta-model: Fig. 2.41 presents the general *SD meta-model*. Here, it becomes clear that the CLD can serve as the beginning for the development of a SFD. This relation is unidirectional since the SFD provides further information which cannot be visualized by the CLD as was described.

Considering the presented general SD model, SD-based simulation models are constructed by the following design rules (Sterman, 2000, p. 204):

1. CLD's core nodes refer to *variables* and *links* having a *polarity*, *delay*, and *loop*.
2. Links should have an unambiguous polarity.
3. Loops are named and numbered.
4. Variable names should be nouns or phrases and should have a clear sense of direction which is preferably positive.
5. Important loops should follow circular or oval paths and information feedback should have curved lines.
6. The diagram should be organized so that crossed lines are minimized.
7. CLDs generally provide high aggregation levels so that diagrams support the overview. They are mainly disaggregated or decompressed if the audience is confused by aggregated visualizations of causal links and intermediate variables.
8. SFD's core nodes refer to *sources*, *sinks*, *stocks*, and *valves*. Further nodes refer to *auxiliary variables* comprising *constants* and *exogenous inputs*.
9. Clouds represent sources and sinks which represent stocks outside the model boundary. Models start with a *source* node and end with a *sink* node so that one can define an area where a process and the simulation is intended to begin and finish.
10. Further, external variables refer to *constants* which are stocks changing little over the time horizon of interest and *exogenous inputs* which are chosen not to be modeled explicitly.
11. Stocks are represented by rectangles.
12. Nodes are directly connected with edges representing a *flow* or *causal link*. Hence, nodes might not be the predecessor or successor of nodes and models consist of alternating elements of *nodes* and *edges*.
13. Inflows and outflows are represented by arrows pointing from and into a stock.
14. There can be no causal links directly in a stock but in regulators.
15. Each edge has only one incoming and one outgoing node.
16. CLD-based and SFD-based models at least consist one atomic flow or a composition of further flows.

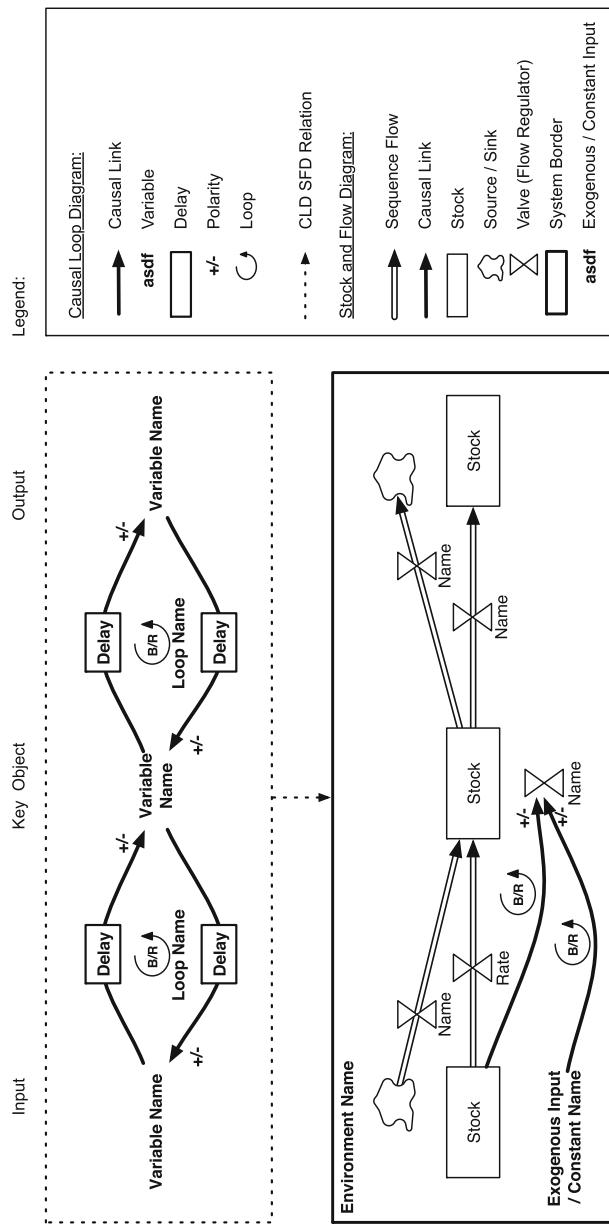


Figure 2.41 Generic System Dynamics meta-model (synthesized from literature)

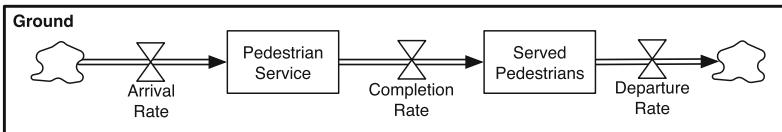
Causal Loop Diagram:Stock and Flow Diagram:

Figure 2.42 System Dynamics-based simulation model example (created to show the strength of the object of investigation)

Example: An example model for a SD-based diagram is visualized in Fig. 2.42. It shows the simulation of three serpentine queue whose context is clarified with Fig. 2.81. Here, pedestrians cross a hall which is not specified any further. For this, they line up at one of three security checks and when they have been served, they are allowed to reach the departure. The figure presented provides its corresponding CLD and SFD. Thus, the example clarifies SD conventions and different modeling perspectives on the same simulation with the aid of different diagram types.

While the SFD clearly visualizes the traveling of customers through the network of stocks (*Pedestrian Service* and *Served Pedestrians*) and flows (*Arrival Rate*, *Completion Rate*, and *Departure Rate*), the traveling starts adding customers to the stock of *Pedestrian Service* which is reduced by the *Completion Rate*. Meanwhile, this rate increases the stock of *Served Pedestrians*. Equivalently, the latter stock is reduced by the *Completion Rate*. The corresponding CLD is shown on top of the SFD. Although technically expressing the same, the CLD complicates the visualization of the physical flow of travelers through the system and their conservation in stocks (Sterman, 2000, p. 211). In the example presented, contradictory causal links represent a unidirectional traveler flow. Further, in CLDs, polarities of causal links are difficult to interpret: Since an increase in the *Service Completion Rate* causes the stock of *Served Pedestrians* to rise above what it would have been otherwise, the polarity is positive. However, in order to characterize the behavior of the stock called *Served Pedestrians*, the comparison of the *Completion Rate* and the *Departure Rate* is essential since the stock will only increase when pedestrians are added faster than they are removed from it. Since concrete numbers are not

specified with the CLD, quite many pitfalls have to be considered here (Richardson, 1986; Richardson, 1997). The SFD therefore is more suitable to specify simulation models according to Def. 16.

Critical appraisal: SD diagrams clearly have their strength in the creation of interpretable and understandable diagrams. Particularly, the SFD can represent simulations models which are adequate to be carried out by simulators because of their concern with values. Hence, modeling language criteria from the simulation domain are essentially satisfied.

Commonly known modeling concepts such as representations about Boolean operators still need to be established in SFDs manually as they are not part of the modeling objects provided. Further, they are not part of the CLD.

Any further modeling language criteria can be satisfied under high implementation efforts as follows: concept specific variables need to be established manually by *regular* and *auxiliary* variables as well as *constants*. Thus, SD non-specific concepts need to be harmonized with the common modeling concept of SD. E.g. modeling entities from the knowledge management domain, such as knowledge carriers, entities from the business process domain, such as customers, and modeling entities that form the NN and ANN domain, such as protuberances, must be modeled by the model creator explicitly in the SFD. Corresponding modeling entities are not provided yet. Additionally, underlying ANN concepts need to be implemented in SD so that concepts can operate under same numerical propagation constraints (Ren, Chai, and Liu, 2005).

Considering the preparation for tools, the SD modeling notations consider basic tool criteria of modeling tools. Exceptional technical concept criteria of modeling tools, such as workflow integrations, interface provision, or augmented reality visualizations are not met anyhow. Simulation tool criteria are met exceptionally with regard to issues about visualizations. Since there are only few approaches that already consider ANN simulations in SD, for e.g. by Ren, Chai, and Liu (2005), ANN functionality is met in a rudimentary manner.

Since the CoNM intends to carry out simulations on behalf of ANN, a neuronal interpretation for stocks has to be established, which is based on the harmonization of neuronal mechanisms with Eq. 2.12 and Eq. 2.13. Particularly, the SFD therefore is suited to be a foundation for the construction of a CoNM because concrete values are considered here. As knowledge perspectives are not considered by system dynamic notations anyhow, a combination with knowledge modeling approaches is ideal, such as the KMDL. Since further ANN concepts are not considered adequately, a combination with ANN programming libraries and tools is suitable.

2.3.3.4 Critical Appraisal of SMLs

Considering the aforementioned simulation modeling approaches and its interim conclusions, it seems that today's simulation modeling approaches show numerous aspects which can be found in simulation modeling taxonomies (section 2.3.2), but they are still at development stage and do not consider ANN techniques:

- None of presented simulation modeling languages considers neuronal processes and neuronal modeling objects. Therefore, an integration of ANN objects with currently available simulation modeling perspectives is missing.
- Only some SD approaches refer to the translation of SD-based models to ANN models (Ren, Chai, and Liu, 2005). As these can neither optimize processes nor create interpretable simulation models or process models, these recommend the retraining of ANN, carry out structural changes or data, an integration of simulation modeling languages with the process domain is missing.
- None of the presented simulation modeling languages consider an *EA concept* (e.g. ARIS) which supports the construction of consistent simulation models being integrated perfectly within a company's structure. This is essential as workflows shall be optimized by neuronal mechanisms.
- None of the presented simulation modeling languages consider phases dealing with neuronal libraries, neuronal tools, neuronal methods, or neuronal concepts. An integration with currently available concepts, tools, or libraries is therefore missing as well.

A research gap becomes visible here.

A synthesis of any simulation modeling approach presented here which will be enriched with ANN modeling objects and required concepts and mechanisms, will support a broad acceptance of the CoNM in corresponding research communities. Further, a synthesis ensures its broad application possibilities. Aiming to not construct impractical modeling approaches, a synthesis must assure that complex models do not reduce the ease of use for any realistic setting (Curtis, Kellner, and Over, 1992).

Hence, for the construction of a CoNM, a compromise must be identified between holism of modeling items required and practicability. In principle, a practical and holistic synthesis must address the following things:

- It must broadly consider simulation method taxonomies (section 2.3.2) and provide modeling objects for its dimensions and manifestations.

- The simulation behavior must be characterizable (Sulistio, Yeo, and Buyya, 2004), which requires modeling items of both deterministic and probabilistic simulation models, providing different kinds of mathematical and computational operations.
- The presentation of simulation time must be supported (Sulistio, Yeo, and Buyya, 2004) so that static and discrete simulation models can be constructed.
- Continuous and concrete values need to be reflected by simulation models (Forrester, 1972; Sterman, 2000; Sulistio, Yeo, and Buyya, 2004).
- Feedbacking structures among simulation model operations need to be considered (Mildenberger and Sauerbier, 1999, p. 24; Fröming, 2009, p. 126).
- The simulation modeling language needs to specify terminating and non-terminating models (Mildenberger and Sauerbier, 1999, p. 24; Fröming, 2009, p. 126).
- Simulation models need to deal with stationary as well as non-stationary systems (Mildenberger and Sauerbier, 1999, p. 24; Fröming, 2009, p. 126).
- Inspired by real time-oriented taxonomies, simulation models need to specify live and time-delayed simulation realizations (Mildenberger and Sauerbier, 1999, p. 24; Fröming, 2009, p. 126).
- Focusing on symbiosis-oriented taxonomies (Aydt et al., 2009a; Aydt et al., 2009b; Tjahjono and Jiang, 2015, p. 823), simulation models need to present the integration of physical systems in simulation environments.
- Further, simulation models need to provide elements for any degree of digitalization, which refers to virtual, hybrid, and real-world simulation components (Lass, 2017, p. 140).
- Simulation models need to provide elements for time-, process-, and event-oriented simulations (Mildenberger and Sauerbier, 1999, p. 31; Fröming, 2009, p. 135).
- Depending on the simulation method specialization (Mildenberger and Sauerbier, 1999, p. 29; Law and Kelton, 2000; Fröming, 2009, p. 1129; Shishvan and Benndorf, 2017), simulation models need to provide method-specific modeling elements.
- A spatial and geographical positioning of simulation processes and operations must be enabled (Grigoryev, 2018, p. 137).
- Positioned simulation model objects need to be processually connected (Grigoryev, 2018, p. 137).
- ANN techniques must be harmonized with the understanding of simulation modeling, process modeling, and knowledge modeling.

The creation of a CoNM will therefore consider these points and it will be based on a practical synthesis of simulation modeling languages presented here.

2.3.4 Conclusion About Taxonomies and SMLs

The conclusion about process simulation is based on the following. First, taxonomic simulation approaches are appraised. Then, simulation modeling languages (SMLs) follow. A simulation modeling-specific facet is subsequently identified.

Taxonomy: Most approaches to classify simulation methods, so-called taxonomies, are based on a set of *dimensions* being operationalized by *manifestations*. The consideration of neuronal perspectives, neuronal modeling objects, and an integration with components from a simulation modeling domain have not been identified in any approach, dimension, or manifestation (see each individual interim conclusion). Since taxonomies were defined to classify attractive simulation modeling elements, which was not aiming for the construction or realization of a CoNM, an appropriate CoNM taxonomy is still missing per se. Hence, taxonomies neither have been evaluated to be the foundation for the CoNM (for e.g. as building blocks on various levels), nor taxonomies have been evaluated to classify elements required by an CoNM so far.

In order to decide whether current taxonomy can at least partly identify a CoNM foundation or which criteria is suited to be part of a CoNM criteria catalog, the following appraises criteria category-wise:

1. Since the CoNM is intended to be carried out in a holistic and general setting, any *dimension* provided by taxonomies is relevant since dimensions focus on aspects which are accepted in the simulation modeling community. When applying accepted concepts from the simulation domain in the ANN process domain, these criteria are highly relevant.
2. Since the CoNM is intended to be carried out in a holistic and general setting, any *manifestation* provided by taxonomies is relevant since dimensions represent design options which are accepted in the simulation modeling community. When applying accepted concepts from the simulation domain in the ANN process domain, these criteria are highly relevant.

Since no approach presents the use of neuronal techniques and concepts which is an essential part of the ANN process domain, none of the provided evaluation frameworks can at least partly identify a CoNM foundation. Further, this implies that

the simulation modeling domain has not been brought together with AI techniques yet. A research gap becomes visible here which requires the following: First is the construction of a CoNM because anything similar is missing so far. Second is a CoNM taxonomy selection system that extends common criteria and can evaluate current taxonomies. Third is an analysis of currently available taxonomies for the use of CoNM-relevant aspects, such as neuronal techniques, specific modeling objects, and common tool criteria.

Simulation Modeling Languages: Most approaches suitable as SML are based on individual sets of *modeling items* and *syntax*, each having a unique modeling strength. The consideration of neuronal perspectives, neuronal modeling objects, and an integration with components from a simulation modeling domain have not been identified in any approach in a satisfactory manner (see each individual critical appraisal). Since simulation modeling languages were defined to perform simulation modeling concepts which do not refer to models coming from a CoNM, an appropriate CoNM simulation modeling language is still missing per se. Hence, SMLs neither have been evaluated to be the foundation for the CoNM (e.g. by providing commonly accepted modeling items and well-defined syntax), nor have SMLs been evaluated to be part of a CoNM so far.

In order to decide whether current SMLs can at least be partly used as a CoNM foundation or which dimension and manifestation is suited to be part of a CoNM evaluation catalog, the following presents aspects category-wise:

1. Since the CoNM is intended to be carried out in a holistic and general setting, any *modeling item* provided by simulation modeling languages is relevant since items focus on aspect, which are accepted in the simulation modeling community. When applying accepted concepts from the simulation domain in the ANN process domain, these criteria are highly relevant.
2. Since the CoNM is intended to be carried out in a holistic and general setting, any *syntax or convention* provided by simulation modeling languages is relevant since syntax and convention focus on aspects which are accepted in the simulation modeling community. When applying accepted concepts from the simulation domain in the ANN process domain, these criteria are highly relevant.
3. Since the CoNM is intended to be carried out in a holistic and general setting, any *modeling concept* being considered by simulation modeling languages is relevant since modeling concepts focus on aspects which are accepted in the simulation modeling community. When applying accepted concepts from the simulation domain in the ANN process domain, these criteria are highly relevant.

Since no SML provided items or syntax for the use of neuronal techniques and concepts in a satisfying manner, which is an essential part of the ANN process domain none of the provided SMLs can serve as CoNM simulation modeling language solely. Further, this implies that the simulation modeling domain has not been brought together with AI techniques yet. A research gap becomes visible here which requires the following: First is the construction of a CoNM simulation modeling language because anything similar is missing so far. Second is a CoNM simulation modeling language selection system that extends common criteria and can evaluate current simulation modeling languages as part of a CoNM foundation. Third is an analysis of currently available SMLs for the use of CoNM relevant aspects such as neuronal techniques, specific modeling objects and common tool criteria.

Facit: Neither simulation method taxonomies and corresponding taxonomy evaluation frameworks consider the integration of the simulation domain and the ANN domain nor SMLs do so. Taxonomic approaches do not provide ANN and Deep Learning criteria. SMLs do not realize the aforementioned criteria. A research gap within the simulation domain becomes evident here. This requires the following three: First is the construction of a CoNM taxonomy, second, the design and demonstration of a CoNM selection and evaluation framework, and third, the implementation and demonstration of a CoNM simulation modeling language.

2.4 Process Optimization

As the structural and behavioral modeling of processes (section 2.1) and the structural and behavioral modeling of knowledge, including their dynamic effects (section 2.2) and their examination in dynamic environments, is enabled by simulations (section 2.3), the testing of any form of changes of process elements, knowledge elements, and simulation elements is enabled by scenario-based simulations in terms of sensitivity analyses. Here, a potential can be identified in the examination of modifications because of neuronal learning procedures. As knowledge-intensive processes might manifest because of the processual behavior of complex structures, such as ANNs, the underlying hypothesis of this contribution is that insights in the working of modeled systems, which follow biological mechanisms of ANNs, can lead to improvements in real systems. So, the following questions about the application of neuronal learning in knowledge-intensive processes can be raised: How can knowledge-intensive business processes be organized in a manner similar to the biological structures of the human brain? Do management principles resemble mechanisms of neuronal networks in machine learning approaches of ANNs? Can

ANN training procedures be controlled by the use of economic principles? In this regard, a potential can be found in the application of concepts of one domain in the other domain and vice versa.

Process optimization is characterized by the following: first, basic concepts and definitions are established; second, different kinds of optimization philosophies are provided based on a literature review. Here, relevant perspectives are identified that will form the foundation for the creation of the *ANN Process Taxonomy* (section 4.1.1) and stand as a standard for model-based process optimizations. Third, prominent organizational learning concepts are examined. These serve to improve any kind of models and implement changes in subjective and objective realities so that they can be tested in any kind of model environment (e.g., simulations) and any kind of original environment (e.g., the real world). Although the process optimization attempts are inspired by the organizational context and primarily intend to optimize business processes, they are not limited to business contexts; rather, they enable process optimizations in general and possess the potential for application in neuronal contexts.

2.4.1 Basic Concepts and Definitions

Basic concepts refer to the definition of the progress of model optimization or what is known as “process optimizing”. Further, learning organizations are defined as a procedure for model optimizations, which is inspired by the KM domain (Section 2.4.1.1).

All together, this stands as the foundation for the following two kinds of investigations: first, for the examination of optimization philosophies as objects of investigation, since these can be interpreted as building blocks for process model and simulation model improvement concepts and consider the target-oriented optimization of business processes in organizations; second, for the examination of organizational learning concepts (as further objects of investigation) to enable the improvement of organizational activities as one of many objects to be improved. Furthermore, concepts provided serve as building blocks for the construction of a CoNM.

2.4.1.1 Optimizing Models

When models are adjusted with the intention to optimize them with regard to a certain *objective*, model optimization takes place. An optimization task is then defined as the search process for the best solution to a system within constraints (Biegler, 2010, p. 1). The optimization objective is determined by an *objective function* that provides a scalar quantitative performance measure and needs to be minimized or

maximized (Boyd and Vandenberghe, 2004, p. 129). So, for optimization problems, we have

$$\begin{aligned} & \underset{x}{\text{minimize}} \quad f(x) \\ & \text{subject to} \quad g_i(x) \leq 0, \quad i = 1, \dots, m \\ & \quad h_j(x) = 0, \quad j = 1, \dots, p \end{aligned} \tag{2.14}$$

in order to describe the search for *optimization variables* $x \in \mathbb{R}^n$ that minimize the objective function $f(x)$ among all x that satisfy the conditions $g_i(x) \leq 0, i = 1, \dots, m$ and $h_j(x) = 0, j = 1, \dots, p$. With this, they are a part of the *search space* $\mathcal{S} \subseteq \mathbb{R}^n$, and if they fulfill constraints, they further become part of the *feasible region* $\mathcal{F} \subseteq \mathcal{S}$ (Michalewicz and Schoenauer, 1996). As variables are modifiable by the model creator *mc* (Def. 1), they can be characterized as *decision variables* and can further be interpreted as *degrees of freedom* in the process (Biegler, 2010).

In Eq. 2.14, $f : \mathbb{R}^n \rightarrow \mathbb{R}$ refers to the objective function to be minimized over the n -variable vector x . Further, $g_i(x) \leq 0$ refers to the inequality constraints following the *inequality constraint functions*, which are $g_i : \mathbb{R}^n \rightarrow \mathbb{R}$, and $h_j(x) = 0$ refers to the equality constraints following the *equality constraint functions*, which are $h_j : \mathbb{R}^n \rightarrow \mathbb{R}$. Since it holds that $m \geq 0$ as well as $p \geq 0$, only if $m = p = 0$, the optimization problem characterizes an unconstrained optimization problem with $\mathcal{F} = \mathcal{S}$ (Boyd and Vandenberghe, 2009, p. 127).

Although, from a mathematical perspective, any change leading to an improvement can be characterized as model optimization, a focus lies on the intentional adjustment that leads to a controllable effect within the original environment so that the following excludes improvements by accident (Gronau and Grum, 2019).

Kinds of Model Optimizations: Faced with various kinds of models, such as process models (Def. 3), business process models (Def. 4), knowledge process models (Def. 14), and simulation models (Def. 16), one can find corresponding kinds of model optimizations, such as process model optimization, knowledge process model optimization, and simulation model optimization.

When processes are operationalized by process models (section 2.1) and process-oriented knowledge models (section 2.2), an optimization means making the best or most effective use of some specified, process- and knowledge-related *set of parameters* without violating some *constraint(s)* (Bussler, 2013). Thus, a *process optimization* attempt can be considered as a high-dimensional parameter tuning problem, whose parameters, for instance, can be visualized by PMLs (section 2.1.3). Further, since the objective function and constraint functions are not linear and not known

to be convex, they even must be characterized as *nonlinear optimization* (Boyd and Vandenberghe, 2004, p. 9).

All activities and decisions that lead to a desired optimization of business processes are designated as *business process optimization* (BPO) (Gronau, 2017). The success of an optimization is measured by key performance indicators (KPIs), that, in business contexts, refer to production times, failure and success rates, the number of produced components, among others. These can individually or jointly be part of the objective function specified (Grum, Bender, and Alfa, 2017; Grum et al., 2018). Considering this, the extent to which a change of the set of parameters specified has lead to an optimization of an objective within a certain scenario can be examined (Gronau and Grum, 2017; Grum and Gronau, 2018a; Grum et al., 2018). For instance, as parameters are represented by knowledge process models, changes can be reflected within the corresponding KML (section 2.2.3).

Suggestions for an optimization can be tested within both the original environment and an arbitrary complex simulation model environment. The latter is particularly beneficial since process changes can be examined in different scenarios, sensitivity analyses prove the functioning of changes, and tests are very cost-effective (Lass, 2017). Here, the course of examination can be operationalized by SMLs (section 2.3.3), so that an optimization can be tested under different sets of simulation parameters, each representing an individual scenario. Accordingly, only if suggestions for improvements have been examined in scenarios can one confirm the extent to which KPIs are improved over scenarios and constraints are violated in a satisfactory manner. Dynamic effects so can be visualized on behalf of SMLs and corresponding parameter visualizations.

Optimization Techniques: Optimization refers to the selection of the right process designs and the application of the most appropriate optimization techniques (Niedermann, Radeschütz, and Mitschang, 2011). When faced with a plethora of optimization techniques in BPO, each being described on different levels of abstraction and on the basis of different process meta-models and application domains, selecting the right technique is very challenging (Niedermann, Radeschütz, and Mitschang, 2011). This challenge is further complicated, because there are no effective methods for generally solving nonlinear optimization problems. So, such attempts demand for the use of several different approaches and compromises, and they are extremely challenging even with as few as ten variables (Boyd and Vandenberghe, 2004, p. 9). Hence, although not best suited by CoNM-based optimizations, current optimization techniques need to be examined with regard to their suitability considering their CoNM foundation.

Besides heuristic tedious activities following the principle of trial and error, the systematic determination of optimal solutions leads to overlapping techniques of separate communities: the mathematical programming community, scientific computing community, community of operations research, and engineering community (Biegler, 2010). Each is further regarded in the following sections of a literature review of the terms “optimization methods” and “optimization techniques”.

1) Mathematical Techniques: Techniques coming from the mathematical community focus on the understanding of fundamental properties of optimization problems and optimization algorithms (Biegler, 2010). Since the existence of solutions, the convergence of algorithms and issues about stability and convergence rates are not focused by the CoNM but rather, mathematical formulations are required for the foundation of the CoNM. Here, mathematical optimization techniques are not considered circumstantially.

2) Computer Science Techniques: Techniques coming from the Computer Science (CS) community are strongly influenced by mathematical properties (Biegler, 2010). However, in distinction from them, CS attempts strongly focus on the implementation of optimization methods and draw attention to their efficient and practical use. Although issues regarding numerical stability, ill-conditioning of algorithmic steps, computational complexity, and performance are not focused by the CoNM, CS-based techniques will be the base for the CoNM construction. Examples of techniques from the CS domain optimizing models, that have been discussed include the following (Zhang et al., 2015):

- Nonlinear Programming (Bertsekas, 1995; Ruszcynski, 2011)
- Simulated Annealing (Kirkpatrick, Gelatt, and Vecchi, 1983; Jeffcoat and Bulfin, 1993; Wegener, 2005)
- Great Deluge Algorithm (Dueck, 1989; Dueck, 1993; Dueck, 2004)
- Metropolis Algorithm (Metropolis et al., 1953; Hastings, 1970; Berg, 2004; Minh and Minh, 2015)
- Threshold Accepting (Dueck and Scheuer, 1990; Althöfer and Koschnick, 1991)
- Ant Colony Optimization (Costa and Hertz, 1997; Dorigo, Di Caro, and Gambardella, 1999; Dorigo and Stützle, 2003)
- Particle Swarm Optimization (Eberhart and Kennedy, 1995; Poli, Kennedy, and Blackwell, 2007; Russell, 2007)
- Hill Climbing (Russell and Norvig, 2003; Russell and Norvig, 2010)
- Stochastic Tunneling (Hamacher and Wenzel, 1999; Wenzel and Hamacher, 1999; Hamacher, 2006)

- RANSAC Algorithms (Fischler and Bolles, 1981; Strutz, 2016)
- Primal-Dual Active-Set Algorithms (Boland, 1996; Hintermüller, Ito, and Kunisch, 2002)
- Evolutionary Algorithms (Back, 1996; Baeck, Fogel, and Michalewicz, 1997; Back, Hammel, and Schwefel, 1997; Vikhar, 2016)
- Artificial Neuronal Networks (ANN; Kennedy and Chua, 1988; Malek and Yash-tini, 2009)

CS-based optimization approaches that issue process models focus on atomic process steps rather than process networks, and an example of the same is atomic chemical process models (Biegler, 2010). Optimizations enabling arbitrary complex networks of atomic process steps are not available as of yet. Reasons for this vary. First, high-dimensional parameter tuning problems suffer from extremely high computational costs. Second, particularly global optimization algorithms require a large number of model runs. This is also related to the very high computational costs during the model tuning process. Finally, high-dimensional parameter tuning problems are mostly simplified by a reduction of parameter dimensions considered before the optimization, which leads to impractical, non-global solutions.

A literature review of the term “neuronal process optimization” identified that particular process optimization approaches considering ANN are rare. Furthermore, such attempts enable parameter optimizations of atomic process steps, such as welding (Casalino et al., 2016), for model-based process control (Westkämper and Schmidt, 1998). Here, attempts intend to use the capability of ANNs to learn the effective interrelations between input and output quantities of processes. The consideration of process networks on behalf of an interpretative modeling has not been realized till date. Advanced attempts construct decomposed ANN meta-models that represent “generic” queuing nodes being interconnected manually (Chambers and Mount-Campbell, 2002a). At the very least, they enable the integrative analysis, but a joint process network construction enabling an interpretative modeling has not realized. This further excludes the use of spatial relations and their effective integration with managerial issues from the real world. This may be because ANNs are considered as black boxes resulting in non-interpretative structures.

3) Operation Research Techniques: Techniques coming from the Operation Research (OR) community focus on the formulation of optimization problems and the development of corresponding solution strategies (Biegler, 2010). Since well-established OR methods mostly refer to well-structured models with linear and discrete elements, and the CoNM intends to consider arbitrary complex, nonlinear models, OR-based optimization techniques are not considered any further.

4) Engineering Techniques: Techniques coming from the process and business process engineering community focus on the application of optimization strategies of often poorly-defined real-world problems, the analysis of solutions, and the reliability and diagnosis of the application of methods (Biegler, 2010). Since experiences are mostly results of failures of the solution method and recovery, engineering experiences and corresponding techniques are highly relevant for the foundation of the CoNM. Examples of techniques from the process engineering community, that have been discussed include the following (Fischermanns, 2013; Toutenburg et al., 2008):

- Business Process Reengineering (BPR; Davenport and Short, 1990; Hammer, 1990)
- Six Sigma (Toutenburg et al., 2008)
- Kaizen (Masaaki, 1986)
- Reference Modeling (Scheer, 1997; Fettke and Brocke, 2016b)
- Service Level Management (SLM; Ellis and Kauferstein, 2012), Process-Oriented Service Level Agreements (PROSA; Knapper et al., 2010), and IT Infrastructure Library (ITIL; Stych and Zeppenfeld, 2008; UK, 2012)
- Failure Mode and Effects Analysis (FMEA) and Failure Mode and Effects and Criticality Analysis (FMECA; Defense, 1949; Defense, 1980)
- Total Quality Management (TQM; Toutenburg et al., 2008)
- Balanced Scorecard (Kaplan and Norton, 1992; Kaplan and Norton, 1993)
- Lean Management Concept (Fischermanns, 2013).

Interim Conclusion: Some authors' deeper analysis of optimization techniques will provide insights into the extent to which the attempts issued by the CoNM already have been considered. Here, the authors are considered from the more abstract, philosophical view of their corresponding *process optimization philosophies* (Bhaskar and Singh, 2014), so that a broad CoNM distribution with a great scientific acceptance is supported. Section 2.4.2 will, therefore, focus on the most prominent attempts from the process engineering domain. Section 2.5 will then focus on ANNs from the CS domain, because of their robust and efficient methods when dealing with imperfections and uncertainties within the original environment.

2.4.1.2 Learning Organizations

A literature review of the term “organizational learning” identified that the concept of learning organizations as a construct of various disciplines yields a great number of definitions, each aiming to boost an organization’s initial state using organizational

learning. Attempts for definitions, *inter alia*, can be found in the fields of social and behavioral science, psychiatry, psychotherapy, organization science, policy science and many others (Visser, 2003). As the CoNM intends to integrate the process and ANN domains and to apply organizational learning strategies within neuronal structures, an adequate definition of learning organization must bring together aspects from a non-machine side, inspired by the process domain, as well as aspects from a machine side, inspired by the ANN domain. Thus, a definition can stand as a foundation for more commonly known concepts.

The following sections present attempts to define different learning organizations with the corresponding cluster-wise understanding of organizational learning (OL), which is inspired by Klimecki and Thomae (1997). Here, clusters have to be seen on an equal footing. The critical appraisal section then reflects knowledge definitions and learning understandings in the context of the construction of the CoNM, so that the interim conclusion section is able to establish a well-founded definition of knowledge, which is adequate for the construction of the CoNM.

Distinct to the understanding of “learning” established in section 2.2.1.1, an “organizational learning” does not follow the traditional principle of *stimulus* and *response* (Zimmer, 1987, p. 47 ff). Hence, OL does not lead to an observable behavioral change in any case. Rather, an organizational learning is interpreted as reflexively dealing with an environment, which results in more complex cognitive structures and the learning system having more complex conceptions about that environment that might refer to inner, non-observable relations.

2.4.1.2.1 Experience-Oriented OL Definitions

Experience-oriented approaches refer to the first attempts dealing systematically with OL. These include efforts by authors such as Cyert and March (1963), March and Olsen (1975), Levitt and March (1988), Levinthal (1991), and Klimecki and Thomae (1997). As such attempts are inspired by behavioral science, organizations are more or less considered rationally acting decision generators. These actions are derived from experiences made by the organization that are transformed in a collection of behavior rules, that are then applied when addressing decision problems (March and Olsen, 1976, p. 67). When decisions are executed, further experiences are created by the organization.

An OL, therefore, means an adaptation of current rules of behavior, when environmental reactions (resulting from an organization’s rule-based behavior) do not match the organization’s experiences. If current rules of behavior are useful, rules of behavior are kept. If they are not useful, they demand extension or modification. So, a learning either refers to the adoption of, first, organizational objectives, second,

attention mechanisms for environmental events, and third, search rules for problem solutions (March and Olsen, 1975; March and Olsen, 1976).

In this setting, learning organizations refer to experience collecting entities that intend to facilitate the experience collecting process of its members, and continuously transforms itself with regard to outdated experiences and rules of behavior.

2.4.1.2.2 Interpretation-Oriented OL Definitions

Interpretation-oriented approaches refer to the most prominent attempts dealing with OL. In this context, one can find works such as the “theory of action” (Argyris and Schön, 1978; Argyris, 1990), the “mental models” (Senge and Sterman, 1992; Kofman and Senge, 1993; MacGill and Slocum, 1996), the “frames of references” (Jelinek, 1979; Shrivastava, 1983; Shrivastava, 1985) and “organization-wide understandings” (Huff and Chappell, 1993). As attempts refer associatively, rather than consequently, to original learning theory of Argyris and Schön (1978, p. 15 ff), approaches tend to perceive organizations as spaces of joint cultural patterns (Klimecki and Thomae, 1997). As interpretations of system-environment-relations, these build the cognitive structures of an organization that are adapted by OL.

An OL, therefore, signifies an adaptation of such spaces and is to be seen as a failure correction of interpretations about the environment. If the environment changes, current interpretations are outdated and need to be changed, so that they fit environmental circumstances. OL is interpreted as the relatively independent learning procedure of organizational members, which results in a change of their cognitive structures and, herein, the included environment interpretations (Klimecki and Thomae, 1997, p. 2). Some consider learning as “single-loop” and “double-loop” learning (Argyris and Schön, 1978), “adaptive” and “generative” learning (Senge, 1990a), “first” and “second-order” learning (Pascale, 1991), and “linear” and “non-linear” learning (Weick, 1991), and some demand phases of an additional “unlearning” (Hedberg, 1981; Klein, 1989; McGill and Slocum, 1993; Nystrom and Starbuck, 1984).

In this setting, learning organizations intend to detect and correct interpretation errors, so that they are able to carry on with present policies or achieve present objectives (Argyris and Schön, 1978). Some authors even consider learning organizations with a design-oriented management approach, which enables an organization to better controlling its evolution on the basis of learning (Senge, 1990b; Pedler, Burgoyne, and Boydell, 1991; Dibella, 1995; Probst, 1994; MacGill and Slocum, 1996).

2.4.1.2.3 Knowledge-Oriented OL Definitions

Knowledge-oriented approaches refer to broadly distributed attempts dealing with OL. Here, one can find work by authors such as Duncan and Weiss (1979), Cohen and Levinthal (1990), Lyles and Schwenk (1992), Slepian (1993), and Nevis, DiBella, and Gould (1995). As attempts are inspired by social systems, organizations are considered as individual entities that provide knowledge bases besides its organizational members. Each provides knowledge about the effectiveness of organizational actions (Duncan and Weiss, 1979) and can be interpreted as cognitive structures that are refined continuously (Klimecki and Thomae, 1997).

An OL, therefore, means the optimization of the joint organizational knowledge base. It refers to the process of adaptation, in which the individual and organizational knowledge bases are changed so that individual and team-based competencies about problem solution and abilities are improved (Thommen and Günther, 2018). If current action results do not match the intended action result, defective knowledge bases become apparent. Then, current observations are abstracted and transformed into knowledge updates that support future matches. So, updates refer to, first, the confirmation of current knowledge, second, the supplementation by new knowledge, and third, the elimination of incorrect knowledge (Duncan and Weiss, 1979).

In this setting, learning organization refers to a joint framework of and for organizational members to develop knowledge (Thommen and Günther, 2018). Here, the intent is to detect and correct incorrect knowledge bases so that its adequate use supports the effective realization of organizational actions.

2.4.1.2.4 Information-Oriented OL Definitions

Information-oriented approaches refer to the most current attempts dealing with OL. Here, one can find work by authors such as Daft and Lengel (1984), Daft and Huber (1986), Huber (1991), Dixon (1992), and Walsh and Ungson (1991). As attempts are inspired by the cybernetic organization theory, organizations are considered as information processing units (Daft and Huber, 1986) that are faced with a stream of information. As such, the better the organizations are because of OL, the better their information processing. Arrangements of information, therefore, can be considered as cognitive structures whose adaptation supports effective information processing.

Here, an OL means an adaptation of an organization's information processing procedures so that relevant information is processed effectively. If the current action outcome does not reflect the expected outcome, the relevant information is either not processed or not processed efficiently. So, a learning either refers to the adaptation of, first, information acquisition, second, information distribution, third, information

interpretation (Daft and Huber, 1986), or fourth, information recapitulation (Daft and Lengel, 1984).

In this setting, learning organizations refer to information processing entities that intend to facilitate the information processing of its members, and continuously transforms itself with regard to ineffective information processing procedures.

2.4.1.2.5 Critical Appraisal of OL Definitions

Despite the development of OL clusters shown above, no approach considers processual *knowledge* and *knowledge processing*, as it was defined in section 2.2.1.1. Further, the use of neuronal mechanisms has not been considered. As the CoNM, by intention models organizations, includes its learning members with the aid of ANNs, it brings in an understanding of neuronal learning. Therefore, the following appraises the extent to which the four aforementioned definition clusters about learning organizations are particular suited to stand as part of the CoNM foundation or must be excluded for this purpose.

Experience-oriented definitions: Even if one might argue that an ANN is a machine-based construct able to carry experiences, so that experience-oriented definitions must be excluded for the CoNM foundation, one can find interpretations allowing for a CoNM foundation. If one assumes an ANN to gain experience during training procedures and its decision rules are coded within neuronal structures, these kinds of definitions are suitable for a CoNM foundation. With their activation by a certain situation, decision rules are applied so that organizational actions can be derived. When current decision rules do not lead to an intended behavior, a change within neuronal structures is realized through ANN learning procedures, which can also be interpreted in the sense of OL.

Interpretation-oriented definitions: Even if one might argue that an ANN is a machine-based construct able to establish interpretations, so that interpretation-oriented definitions must be excluded for the CoNM foundation, one can find interpretations allowing a CoNM foundation. Say, one assumes an ANN is facing sensory input from the original environment, particularly, the first levels of an ANN, which deal with the building of interpretations. Since these can be interpreted as mental models within the subjective reality of an ANN, these kinds of definitions are suitable for a CoNM foundation. Further neuronal levels will take part in producing expectations and, occasionally, decisions with the aid of these mental models. Here, ANN learning procedures allow the adaptation of wrong interpretations such that the

subjective reality of the model creator and the original environment match, which carries out learning in the sense of OL.

Knowledge-oriented definitions: Even if one might argue that an ANN is a machine-based construct able to develop knowledge, so that knowledge-oriented definitions must be excluded for the CoNM foundation, one can find interpretations allowing a CoNM foundation. If one assumes an ANN to carry knowledge which is coded within the ANN structures, activation of the ANN leads to the consideration of knowledge so that organizational actions can be derived. Compartments of substructures of the ANN can be considered as separate knowledge bases. Here, learning guarantees the optimization of knowledge bases so that knowledge is developed by ANN procedures in the sense of OL. Hence, such definitions are suitable for a CoNM foundation.

Information-oriented definitions: Even if one might argue that an ANN is a machine-based construct able to intentionally distribute and process information, so that information-oriented definitions must be excluded for the CoNM foundation, one can find interpretations allowing a CoNM foundation. If one assumes an ANN to be a collection of information processing units, then a neuronal learning refers to the optimization of ineffective information processing procedures by ANN learning procedures. For instance, this applies along with weight adjustments within the network of neurons. As such, the information processing of arbitrary complex processing units is improved. In the case they represent organizational units, adaptation procedures are in the sense of OL. Hence, these kinds of definitions are suitable for a CoNM foundation.

Since none of the presented kinds of learning organizations or definition clusters of OL have to be excluded for the construction of a CoNM, the following intends to establish a joint definition so that a possibly broad acceptance of the CoNM is supported.

2.4.1.2.6 Interim Conclusion

Given the model definition (Def. 1) and process definition stated before (Def. 3), learning organizations and procedures of OL need to be modeled as they are an essential part of process models. Considering the knowledge definition (Def. 13), learning organizations consequently consider knowledge in order to carry out organizational actions and learning. Hence, this research defines models of learning organizations as follows:

Definition 19 (Learning Organization).

A learning organization is a model of an organization of the original environment that serves as a joint reference framework of and for organizational members. It consequently carries out an organizational learning and, therefore, it intends to, first, detect and correct errors in order to either carry on present policies or achieve its present objectives, and second, to continuously update organizational knowledge bases, experience-based decision rules, interpretation patterns, and information processing procedures so that detection and correction competences are enriched.

Since both, *learning organizations* and *organizational learning* are recognized as models in the sense of Def. 1, constructive and formalized operation with these two abstract and intangible entities is enabled, which includes the description, simulation, analysis, design, and evaluation on the basis of more or less subjective interpretations of, limited perceptions of an object system by, and purpose-related design decisions of the model creator *mc*. It is not issued by whom the model is constructed. This allows CoNM mechanisms, human-based, and machine-based model creators to stand side by side and work individually and cooperatively.

As the definition does not specify the term “organizational member” any further, it can stand for the same or different human-based process participant(s) (coming from the process context), machine-based business process participant(s) (coming from the business process context), neuronal representative(s) (coming from the ANN context), or any other modeled entity of knowledge-related *actual entities*.

Since the definition does not specify the perception perspective of “errors”, its determination is left open to whomsoever detects an error. So, changes can be initiated at any organizational level and from any member, which allows top-down and bottom-up initiatives.

Additionally, it is not specified any further which kind of error the definition deals with. Thus, it can refer to experience errors, interpretation errors, knowledge errors, and even information processing errors. Since all of them can be reflected within process models, if they are processually interpreted as *actual entities*, various kinds of OL are reflected by the definition presented.

When the definition speaks about “present policies” and “present objectives”, a target-oriented improvement is issued. Although an OL does not necessarily lead to an measurable behavioral change (Zimmer, 1987, p. 47 ff), at least inner structures of organizational knowledge bases, experience-based decision rules, information processing procedures, and interpretation patterns are affected, which can be measurable by KPIs. As others are not measurable, the definition speaks about a “competence enrichment”.

Since *learning organizations* and *organizational learning* are recognized as processes, they are allowed to be interrelated with EA (Def. 7). This means that, first, they are embedded in data, hardware, software, and communication resources, which means that the knowledge process model is fed with EA data, is carried out on EA hardware and software, and considers EA hardware, software process participants, and available communication structures. Second, they are allowed to refer to EA activities, which refer to the organization of data, hardware, software, and communication resources. Here, the organization of organizational entities following biologically-plausible mechanisms of the brain are enabled.

All together, on the basis of the presented definition of learning organizations, organizations can be considered as dynamic phenomenon and visualized over time. Here, the CoNM is intended to consider processual circumstances as well as organizational learning. Since this consideration may be visualized on behalf of process modeling languages (introduced in section 2.1.3), knowledge modeling languages (introduced in section 2.2.3), and simulation modeling languages (introduced in section 2.3.3), a simulation, knowledge, and process modeling can be considered jointly, which is the foundation of efficient knowledge management (introduced in section 2.2.2). If here, neuronal mechanisms are used in order to create process models, the learning organization can be observed by analogy with biological organization principles of the human brain. So, the CoNM constructed will be the foundation for the OL mechanisms to the biological world and mechanisms from the latter to the first world.

The deeper analysis of some authors' works on OL will provide insights into the extent to which attempts issued by the CoNM have already been considered. Here, the authors are chosen from the cluster of the interpretation-oriented definitions, because it shows the widest distribution, the greatest scientific acceptance, and fits best with regard to the model construction philosophy. This demands a model creator that constructs interpretations of the original environment and adapts during the process of model construction so that mental models of the subject reality fit. Section 2.4.3 will focus on the most prominent attempts of an interpretation-oriented OL.

2.4.2 Optimization Philosophies

In accordance with the model optimization understanding presented in section 2.4.1.1, organizational principles focusing on the efficient improvement of process models are referred to as *process optimization philosophies* (POPs) (Bhaskar and Singh, 2014). These are closely connected with operational questions about the

redesign of models, the use of knowledge, their documentation, and their successful implementation within the individual process level or the entire organization (Grum et al., 2019).

Selection of representatives: A literature review of the term “process optimization” identified various concepts of *philosophical attempts* providing more or less sophisticated approaches adequate for an improvement of process models. Authors mostly refer to the following three kinds of optimization philosophies (Dumas et al., 2013; Gronau, 2016, p. 196): the complete redesign, issued in the first sub-section, the heuristic-based improvement, issued in the second sub-section and the process adjustment on behalf of reference models, which is part of the third sub-section. A last sub-section then critically appraises optimization philosophies in terms of process optimization.

In general, any POP concept presented provides the following two CoNM potentials: first, a potential for being a foundation for the CoNM; and second, a potential for the inclusion of organizational concepts on the basis of neuronal mechanisms. On behalf of the CoNM, the transfer of organization principles among both domains shall be enabled. As it was possible to identify, on behalf of the CoNM, how processes can be improved in organizations following biologically-plausible and centuries-old, evolutionary optimized principles, economic potential can be assumed to be uncovered. Further, as it was possible to identify, on behalf of the CoNM, how dynamic activities can be reflected in neuronal structures following organizational principles, a potential regarding the power of ANNs and their interpretability can also be uncovered. The following examines prominent work wherein biological principles of the brain have been considered. The contribution, therefore, selects POP concepts according to the following criteria:

1. Faced with the process optimization concept established in section 2.4.1, only research considering the improvement of *process models* have been considered. This puts a focus on a process improvement.
2. POPs having a broad acceptance in the process optimization community.
3. POPs being applied in at least one scientific publication.

Adequate examination level: In order to satisfy requirements for an adequate dealing with objects of investigation (OoI) in the terms of SLR (Levy and Ellis, 2006), which corresponds to the application of Bloom’s taxonomy (Bloom et al., 1956), each KM concept is described by the following categories:

- The short *abstraction*, which serves as a kind of introduction to POPs on the same abstract level. Therefore, it is presented at the very top of each sub-section.
- Its individual *description*, which characterizes the individual OoI.
- *Side facts* issuing the OoI-specific development and historic evolution so that its standing within the scientific community becomes clear.
- The provision of a *model*, which visualizes POP activities and its underlying context. This is completed by *explanations* clarifying how the model of the OoI is applied so that a concrete optimization can be carried out.
- An *example*, which visualizes the application of the OoI and underlines management strengths of the OoI.
- Finally, the *critical appraisal*, which evaluates if and how the individual POP concept is suited to stand as part of the foundation of the CoNM.

So, according to Bloom's taxonomy, categories considered for any POP concept correspond to the levels of *knowing* and *comprehending* if OoIs are described and side facts are presented. Further, to the level, *applying* as an abstract POP model is applied and an example is presented, *synthesizing* as a common level of understanding is worked out, *analyzing* and *evaluating* as common analysis criteria are issued, and the POP concepts are regarded with respect to the CoNM.

2.4.2.1 Business Process Reengineering

Business process reengineering (BPR) is a management concept whose focus lies on a reengineering of current processes to devise new ways of organizing business objects, such as tasks, people, and IT systems (Chen, 2001; Hess and Schuller, 2005).

Description: Although the presence of a commonly agreed upon definition of BPR is argued in literature (Meel, Bots, and Sol, 1994; Peltu, Clegg, and Sell, 1996; Mac-Intosh and Francis, 1997; Bhaskar and Singh, 2014), widely accessed definitions refer to the concept of BPR as the fundamental over-thinking of as-is processes, such that dramatic improvements in critical, contemporary performance measures are realized (Hammer and Champy, 1993). Mostly, this is connected to far reaching changes and even a completely new design of processes and the organization itself. Here, process improvements are designed as if the organization was built anew and current knowledge and state-of-the-art techniques are additionally considered (Grum et al., 2019). This includes the analysis and design of workflows among organizations, so that the maximum effectiveness is realized (Davenport, 1993).

Central BPR tenets refer to the several statements (Vidgen et al., 1994). First, changes are radical and concern any organizational structure. Second, current

assumptions are challenged so that being trapped by the way things are currently done is avoided. Third, changes support the establishment of a process and goal orientation. Since each process is considered as a part of a process chain affecting the customers, any process needs to add value to what a customer wants. Fourth, an organizational restructuring is accepted, which is not limited by an organization's border. Rather, it considers an organization as a fluid mix of interests within a network of companies. Finally, the use of contemporary technologies is supported. Since these enable creative solutions, innovative experiments support the identification of maximum effectiveness.

Since improvements are carried out following a top-down strategy, BPR results provide the several characteristics (Hammer and Champy, 1993). First, decisions are *decentralized* because hierarchical structures are flattened in favor of a process-oriented structure. Second, process step sequences are *reorganized* and different *process variations* are examined. Third, the *localization* of working content is organized meaningfully, which puts a focus on the assumption-free reorganization of work. Finally, the need for *control* and required *coordination efforts* are reduced. For instance, as centralized contact points for customer requests are established, an effective and self-responsible organization is supported.

Side facts: The original concept of BPR can be traced back to management theories of the twentieth century authored by Frederick Taylor in 1980 and Henri Fayol in the early 1900s (Chen, 2001; Radhakrishnan and Balasubramanian, 2008; The Financial Times, 1994). Although initially drawing academic and industrial attention as a result of two papers by Michael Hammer on reengineering (Hammer, 1990) and Thomas Davenport on business process redesign (Davenport and Short, 1990), the field of BPR emerged when the key books of Davenport (1993) and Hammer and Champy (1993) were released (Hess and Schuller, 2005).

Since then, BPR has widely been seen as a solution for economic crises, since the organization efficiently adapts to environmental circumstances on behalf of BPR projects. For instance, in the 1980s, this referred to financial reengineering being faced with financial crises, in the 1990s, to technological reengineering being faced with new IT, and similar fundamental redesigns till date because of more global business environments and the more complex taste of customers (Chen, 2001).

Meta-model: The BPR meta-models are presented in Fig. 2.43. It shows the typical six steps of a reengineering life cycle including the corresponding activities (Guha, Kettinger, and Teng, 1993; Gong, 2013, p. 176; Kiran, 2016, p. 410).

First, the *envisioning* of a BPR project is secured and reengineering opportunities are exhausted. By identifying enabling technologies, the project is aligned with the

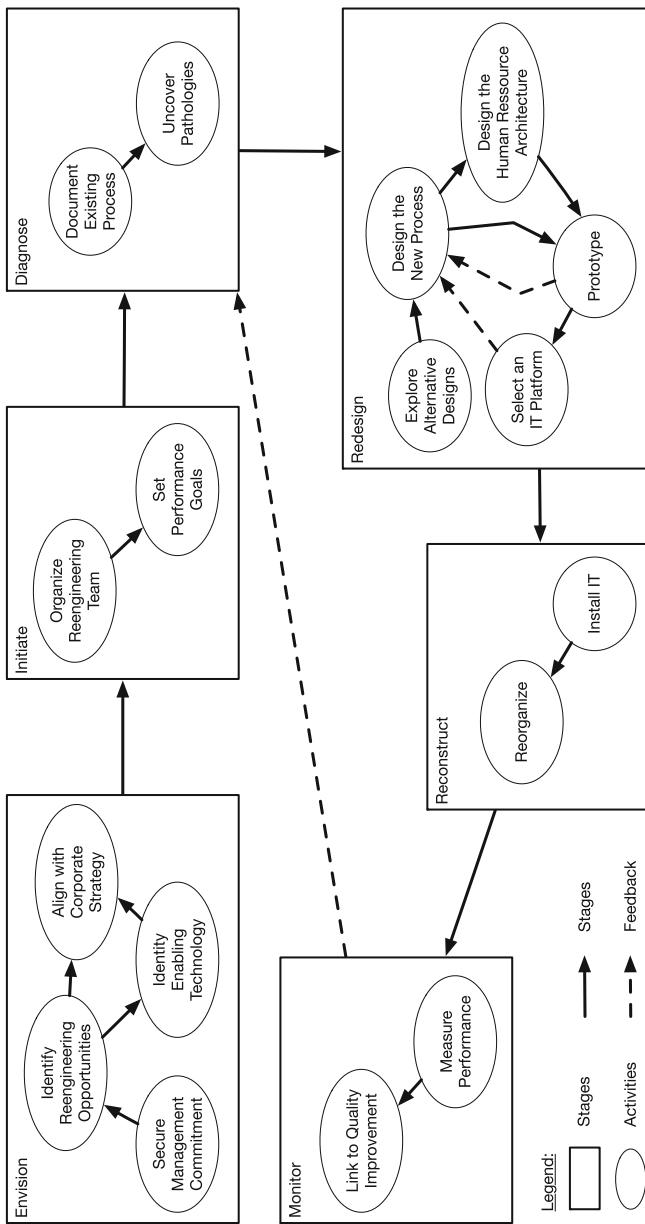


Figure 2.43 Generic Business Process Reengineering Meta-Model (synthesized from literature)

existing corporate strategy. Second, the *initialization* of the BPR project is realized. Here, the team is put together and performance goals are identified whose completion determines the project's success. Third, the *diagnose* stage uncovers current pathologies on behalf of as-is process documentations. Fourth, during the *redesign* stage, alternative designs are constructed with the aid of prototypes and IT platform selections and designs of new processes are part of feedback iterations. Fifth, at the *reconstruct* stage, IT is installed and reorganization measures are realized. Finally, during the sixth stage, the *monitor* stage, the performance of the BPR projects is evaluated and a link to the quality improvement is established.

Example: An example application for the BPR refers to the business process reengineering of a company's car manufacturing process on behalf of interventions focusing on knowledge velocities (Gronau and Grum, 2019). During the envision stage, specifically, knowledge-intensive production steps are identified as reengineering opportunities. The use of AR glasses is identified as enabling technology that is in harmony with the corporate strategy. When the reengineering team was composed, a reduction of 10% of the failure rate was set so that failures which were made during the production process are issued. The diagnose stage showed that failures were mainly made by inexperienced workers. So, at the redesign stage, a focus is laid on the reorganization of production steps by augmenting activities stepwise. Particularly, inexperienced workers were guided by detailed instructions so that typical failure possibilities throughout the entire process are prevented. After new process designs have been realized in the reconstruct stage, their performance is measured in reality. Because of the link to quality improvement, feedback loops are established and a CIP is enabled (section 2.4.2.2).

Critical appraisal: As the BPR can be interpreted as a generic management approach, modeling objects of any modeling approach can be issued. So basically, modeling language criteria from all modeling perspectives (e.g. business process domain, knowledge domain, simulation domain, NN and ANN domain) can be affected by BPR projects. Since these are not issued explicitly, this demands advanced implementation efforts.

Since no ANN optimization is issued by BPR approaches, the marriage of the business process domain (BPR side) and the ANN domain (neuronal perspectives) is not realized at this stage. Corresponding ANN-based modeling language criteria therefore are not met at all. Further, corresponding ANN tool criteria are not met anyhow.

Modeling and simulation tool criteria can be met at a rudimentary level, because the BPR issues the process variant wise examination and includes the monitoring

and KPI-wise performance evaluation e.g. But in particular, technical tool criteria are not met anyhow.

As the BPR generally enables the fundamental reorganization of process models, it is attractive for harmonizing with the radical change of ANN structures during training phases. Similarly, radical changes in ANN structure because of weight adjustments are attractive to be harmonized with BPR-based changes. So, BPR-approaches will serve as the foundation for the construction of the CoNM.

2.4.2.2 Continuous Improvement Processes

Continuous improvement process (CIP) is a management concept whose focus lies on a never-ending improvement of processes and products in small steps (Gronau and Grum, 2017; Grum and Gronau, 2018a).

Description: Following Masaaki, key principles include a *feedback* mentality and processes that are reflected continuously (Masaaki, 1986). The everlasting search for *efficiency* demands for the identification and improvement of suboptimal processes such that waste is reduced and eliminated. Further, the emphasis lies on continual steps rather than drastic changes, which is connected to the key principle of *evolution*. Corresponding management concepts follow cyclic procedures and can be found in numerous variations: Shewhart Cycle (Shewhart and Deming, 1939), Deming Wheel (Deming, 1950), a second Shewhart Cycle (Deming, 1986), plan-do-study-act (PDSA) Cycle (Deming, 1993), plan-do-check-act or plan-do-check-adjust (PDCA) Cycle (Moen and Norman, 2006), and a second PDCA Cycle (Ishikawa, 1985). In general, those concepts contain planning activities (*plan*). Afterwards, the process is implemented and carried out (*do*) and Feedback is collected and compared to a planned output (*check*). Then, changes are implemented constantly or revised before this cyclic procedure is resumed. The feedback collection can gather process data coming either from a process realization of the real world or from simulated processes. Since improvement ideas are generated during the process realization stage and a focus lies on single processes, improvements are carried out bottom-up.

Side facts: Originally, the CIP was inspired by the Japanese living and working philosophy called *Kaizen*. Western economy then overtook this principle of never-ending changes and laid a focus on quality increase, cost reduction, and time efficiency issues. Therefore, it is an integral part of quality management concepts (Fischermanns, 2013).

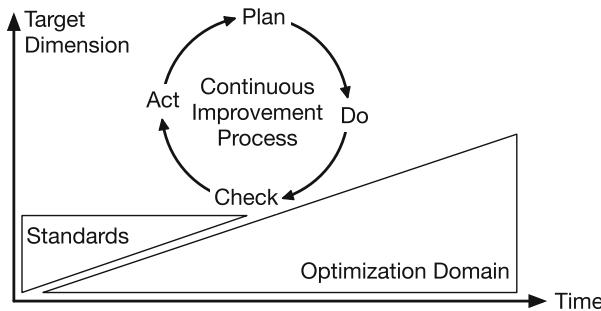


Figure 2.44 Generic Continuous Improvement Process Meta-Model (synthesized from literature)

Meta-model: According to Gmech et al. (2015), Fig. 2.44 provides a general CIP meta-model. Applying the circular CIP, the optimization metaphorically refers to the laborious rolling up of a ball within the corresponding *optimization domain*. While carrying out these small steps, an object thus improves with regard to a certain *target dimension*. Through the iterative CIP realization, the optimization object improves continuously. *Standards* are useful for any iteration, since they can serve as an attractive initial position of the CIP or as fall back positions for suboptimal changes.

Example: An example application for the CIP refers to the consideration of *restructure process changes* such as outsourcing, condensation, parallelization, acceleration, and transfer that all intend to raise the process realization velocity (Gronau, 2011, p. 28; Gronau, 2016). During the planning phase, for instance, current process models are analyzed with regard to the successful application of knowledge. Since processes do not profit from the use of experts and show shortcomings in qualitative results, the change of current process models is designed. In this example, this refers to the transfer of knowledge intensive tasks to experts of the corresponding knowledge domain. Required changes are then carried out in the doing phase of the CIP. The extent to which the changes are successful and have led to an optimization of the initial situation is examined in the check phase. If changes have not led to an improvement, management interventions are performed in the act phase. Since changes have improved the initial situation in this example, the issue of how to

take process models to the next level is raised, for e.g., by the setting up of new goals and actions or more detailed studies (Remmen, 2007). Then, on behalf of the improved situation and knowledge gained from the current CIP iteration, the next CIP iteration is started.

Critical appraisal: Since the CIP is a management approach rather than a modeling concept, individual modeling objects can be examined for improvements on behalf of the CIP. Hence, modeling criteria from all modeling perspectives (e.g., business process domain, knowledge domain, simulation domain, NN and ANN domain) can be considered in CIP iterations.

As a management approach that affects process models and can be tested by simulation tools, basic modeling and simulation tool criteria can be satisfied as a repetitive modeling and optimization on behalf of the simulation results can be carried out easily. As some criteria are not affected by the management concept, such as the database integration, workflow integration, or interface provision, these are not suited for supplementation.

Furthermore, the CIP as an optimization mechanism does not issue neural optimization mechanisms. This was first issued by Grum and Gronau (Gronau and Grum, 2017; Grum and Gronau, 2018a). Therefore, the CIP generally does not satisfy ANN tool criteria. As such, it has not been prepared for an integration with the ANN domain till date. Since the CoNM aims to do so, the CoNM intends to stand as part of the foundation for the integration of CIP, focusing on processes in ANN tools.

As the CIP generally enables the stepwise optimization of process models, it is can be ideally harmonized with the stepwise optimization of ANN. CIP-based approaches, therefore, serve as part of the foundation for the construction of the CoNM, and the CoNM follows such ideas to consider ANN mechanisms for process optimizations (Gronau and Grum, 2017 and Grum and Gronau, 2018a).

2.4.2.3 Reference Modeling

Reference modeling (RM) refers to an optimization approach in which *reference models* are considered as construction results that are intended to be reused. This includes their adaptation and extension so that they ideally fit to the corresponding application context (Gronau, 2016; Fettke and Brocke, 2016b).

Description: In the literature, various definitions can be found issuing *reference models* (Brocke, 2003; Fettke and Loos, 2004b; Fettke, 2007; Fettke and Loos, 2007). Although a commonly accepted term has not been established yet, a convergence can be identified which refers to the reuse-oriented understanding (Fettke

and Brocke, 2016a). Building on this understanding (Alpar et al., 2002; Brocke, 2003; Becker, Delfmann, and Knackstedt, 2004; Becker and Knackstedt, 2004), it is defined as follows:

Definition 20 (Reference Model).

A reference model is referred to as a model of an arbitrary object out of the subjective or objective model reality that was either intentionally or actually developed by the reference model creator to be reused for the construction of, first, further reference models by the reference model user, or second, further application models by the application model constructor, so that the relation of reference model and application model is characterized by the reuse of modeling objects or content of the reference model at the construction of modeling objects or content of the application model.

Hence, a reference model can be considered to represent an ideal for a certain context since, here, it has been proven in reality. It, therefore, can be interpreted as a recommendation for comparable situations. So, the extent to which the model can be overtaken entirely, must be adapted to the corresponding situation or is worse than the current model used in that context must be evaluated. Since the definition does not exclude ANN process models, as they will be created by the CoNM, models created by ANN mechanisms can also be regarded as reference models. Further, the definition allows the flexible exchange of models among domains. For example, neuronal processes might be useful to stand as reference for business processes and knowledge-intensive business processes might be useful to stand as reference for neuronal processes and so on.

Following the metaphor of reference models being applied at the construction of application models, different types of model creators are introduced, each having an individual subjective reality (Fig. 2.5). This makes the concept of reference models particularly attractive for human and non-human experts considering multi-perspective and numerous collections of influence factors. ANN techniques are particularly promising in this context as they can deal with a great number of dimensions.

Since the reference model definition is in harmony with Def. 3 (process models), Def. 4 (business process models), Def. 14 (knowledge process models), Def. 16 (simulation models), and Def. 32 (neuronal process models), the definition enables different kinds of reference models, each optimizing the corresponding type of model.

In general, each kind of object documentation can be considered as a reference model and one can refer to object generations and reuse variations in terms of

product generation engineering (PGE; Albers et al., 2014; Gronau and Grum, 2019). While each ontological level might establish its own reference models being handled as *standard* within a certain range, public collections implicitly provide reference models following *best practices* or *common practices* (Fettke and Brocke, 2016a).

Side facts: Initially, reference modeling came up in the domain of business informatics and aimed at the repetitive use of data and process models (Scheer, 1997). Since the initial RM could demonstrate in a wide range of contexts, such as industrial companies (Scheer, 1997), trading companies (Becker and Schütte, 2004), software companies (Fettke and Brocke, 2016b), and various sectors (service: ITIL, quality: Six Sigma, and so on), the RM has been established as a proper research domain (Fettke and Brocke, 2016b) and now considers more than data and process models. So, to date, domains of public model collections have been diversified (Szigeti et al., 2014; Sarma et al., 2018) and different design fields and methodological varieties have been established (Fettke and Brocke, 2016b). This includes corresponding certification possibilities, such as those offered by the Project Management Institute (PMI), International Project Management Association (IPMA), and PRINCE2, among others. Further, this includes different kinds of conceptual optimization principles such as rules of process design (Gronau, 2016).

Meta-model: Building on the work of Fettke and Brocke (2016b), Fig. 2.45 provides a general RM *meta-model* and thus, clarifies the relation between reference models and application models as well as the relation between the constructive modeling and applicative modeling.

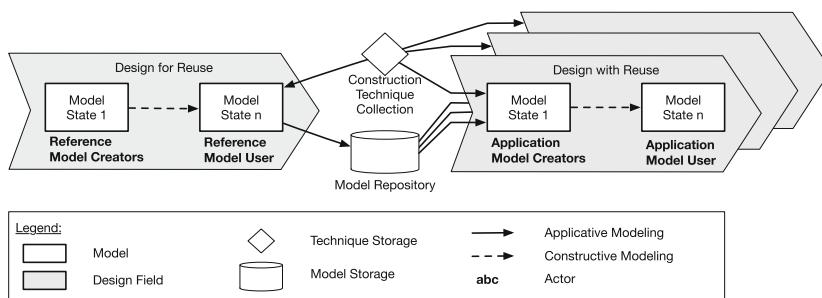


Figure 2.45 Generic Reference Modeling Meta-Model (synthesized from literature)

While the reference models are considered by reference model creators and users in the design field as *Design for Reuse*, application models reusing modeling objects and content of reference models are considered by application model creators and users in the design field called *Design with Reuse*. Although across different design fields, all consider commonly known *construction techniques*.

As a *model repository* stores different kinds of models and provides models for both kinds of design fields, the applicative and constructive modeling can be differentiated by the following: constructive modeling relies on the characterization of the application domain, the construction of modeling elements, and subsequent evaluation and maintenance, while applicative modeling relies on the identification of adequate reference models and their adaptation as well as their integration with further models and their application (Schütte, 1998; Schlagheck, 2000; Brocke, 2002; Fettke and Loos, 2007).

Example: The RM can be exemplified with the customization of an Enterprise Resource Planning (ERP) System. As the software vendor has constructed reference models for trading companies, these are stored in industry-specific model repositories. Here, reference models are implemented in the generic ERP system. In more or less laborious customer projects, each representing an individual *Design with Reuse* field, reference models are analyzed with regard to the question if they address customer-specific processes. The industry-specific contextualization supports the effective reference models selection. Then, application model creators construct process models that are oriented to selected reference models but consider the specific situation of the customer. With the aid of application models constructed, the ERP software is configured so that the generic ERP system considers current application models. Particularly for small and medium-sized enterprises (SMEs), the orientation to reference models is effective as they contain proven knowledge about the effective design of processes. In considering these reference models, knowledge spills into the SMEs and thus, optimizes the SMEs' initial situation.

Critical appraisal: As an generic optimization concept enabling the improvement of different kinds of models, the RM does not issue modeling entities explicitly. Generally, any modeling object can be issued to be part of a reference model. Hence, modeling language criteria from all domains, particularly the process domain, business process domain, knowledge management domain, simulation domain, and ANN domain, can be satisfied through high implementation efforts. Since the RM has not been issued by the ANN domain, corresponding modeling language criteria can also be satisfied by very high implementation efforts.

As a manageable, collective optimization approach, the RM basically satisfies modeling and simulation tool criteria. Particularly, a multi-user support, publication of reference models, database integration, and version control only need small configuration efforts since they are key elements in RM. However, elements concerning the monitoring and visualization are not met. Further, the ANN tool criteria are not considered from an RM perspective as of yet. Although individual concepts are provided in various forms, some by scientific publications, some by software applications, some by programming libraries, and some by public collections (Szigeti et al., 2014; Sarma et al., 2018), a reuse in the sense of RM is not realized anywhere.

Since the RM generally enables the reuse of models created, it is an attractive component of a foundation for the CoNM. First, it enables the flexible reuse of models among domains and thus, supports the ANN process domain. This includes the reuse of process standards for the construction of ANN models, as well as the reuse of ANN processes as standards for the construction of process models. Second, it enables model construction independent from the nature of the model creator. So, human-based and non-human-based creators are allowed to stand side by side, which is a requirement for the CoNM.

2.4.2.4 Critical Appraisal of Philosophies

Faced with the aforementioned process optimization philosophies and its interim conclusions, it seems that today's philosophical approaches are able to consider numerous ways to optimize models so that a certain criteria is improved, but they are still immature and do not consider ANN techniques. This is further elaborated upon in the following:

- None of the presented process optimization approaches considers neuronal modeling objects. Therefore, an integration of ANN objects with currently available optimization concepts is missing. Further, an integration of philosophical optimization objects with currently available ANN-based optimization concepts is also missing.
- None of presented process optimization approaches considers neuronal processes. Therefore, an integration of ANN mechanisms with currently available optimization approaches is missing. Further, an integration of philosophical concepts with currently available ANN-based mechanisms is also missing.

Therefore, a research gap becomes visible here.

A synthesis of any philosophical process optimization approach presented here, which will be enriched with ANN modeling objects and the required concepts

and mechanisms, will support a broad acceptance of the CoNM in corresponding research communities. Further, a synthesis ensures its broad application possibilities. A synthesis must assure that complex models do not reduce the ease of use for any realistic setting, instead of constructing impractical modeling approaches (Curtis, Kellner, and Over, 1992).

Hence, for the construction of a CoNM, a compromise must be identified between the holism of modeling items required and their practicability. In principle, a practical and holistic synthesis must address the following:

- It must look broadly for fundamental changes considering a reengineering (section 2.4.2.1) and provide modeling objects for its focus. Here, radical process changes can be inspired by radical changes in neuronal structures, which result in radical shifts in neuronal behaviors.
- It must look broadly for incremental changes considering cyclic and continuous changes (section 2.4.2.2) and provide modeling objects for its focus. Here, incremental changes can be inspired by lightweight adjustment of neuronal structures, which results in slight shifts in neuronal behaviors.
- It must look broadly for the reuse of models outside of current knowledge bases, which includes construction techniques and model repositories (section 2.4.2.3) and provide modeling objects for its focus. Here, standards and best practices can be inspired by the reuse of pre-trained neuronal structures, standard training data sets, and commonly accepted training and test proceedings.
- ANN techniques must be harmonized with the understanding of process optimization. This includes phases, responsible authorities, and roles, among other factors.

The creation of a CoNM will, therefore, consider those points and it will be based on a practical synthesis of philosophical process optimization approaches presented here.

2.4.3 Organizational Learning

In accordance with the understanding of learning organizations presented in section 2.4.1.2, learning principles focusing on the efficient improvement of models in organizations are referred to as *organizational learning* (OL). These are closely connected with operational questions about the improvement of models by the consideration of knowledge, their documentation, and successful implementation of changes within the individual process level or the entire organization.

Selection of representatives: A literature review of the term “organizational learning” identified various concepts of *optimization attempts* providing more or less sophisticated approaches adequate for an improvement of process and simulation models. Authors mostly refer to the following three kinds of OL concepts (Bateson, 1972; Argyris and Schön, 1978; Visser, 2003; Gronau, 2016): the simple Single-Loop learning, issued in the first sub-section (section 2.4.3.1), the more complex Double-Loop learning, issued in the second sub-section (section 2.4.3.2), and the Deutero-learning, which is part of the third sub-section (section 2.4.3.3). A last sub-section then critically appraises OL approaches in terms of process optimization.

In general, any OL concept presented provides the following two CoNM potentials: first, a potential for being a foundation for the CoNM; and second, a potential for the inclusion of organizational concepts on the basis of neuronal mechanisms. On behalf of the CoNM, the transfer of organization principles among both domains is enabled. As it was possible to identify, on behalf of the CoNM, how organizational learning processes can be improved in companies following biologically-plausible and centuries-old evolutionary optimized principles, it can be assumed to possess an economic potential. Further, as it was possible to identify, on behalf of the CoNM, how OL activities can be reflected in neuronal structures following economic principles, the powerfulness and interpretability of ANN can be assumed. The following examines prominent works to the extent that biological principles of the brain have been considered. The contribution, therefore, selects OL concepts according to the following criteria:

1. Faced with the process optimization conception established in section 2.4.1, only authors considering the improvement of *process models* have been considered.
This puts a focus on a process improvement.
2. OL concepts having a broad acceptance in the process optimization community.
3. OL concepts being applied in at least one scientific publication.

Adequate examination level: In order to satisfy requirements for adequate handling with objects of investigation (OoI) in the terms of SLR (Levy and Ellis, 2006), which corresponds to the appliance of Bloom’s taxonomy (Bloom et al., 1956), each KM concept is described by the following categories:

- The short *abstraction* serves as a kind of introduction to OL concepts on the same abstract level. Therefore, it is presented at the very top of each sub-section.
- Its individual *description* characterizes the individual OoI.
- *Side facts* issuing the OoI-specific development and historic evolution so that the standing within the scientific community becomes clear.

- The provision of a *model* visualizing OL activities and its underlying context. This is completed by *explanations* clarifying how the model of the OoI is applied, so that a concrete optimization can be carried out.
- An *example*, which visualizes the appliance of the OoI and underlines management strengths of the OoI.
- Finally, the *critical appraisal*, which evaluates if and how the individual OL concept is suited to stand as part of the foundation for the CoNM.

So, according to Bloom's taxonomy, categories considered for any OL concept correspond to the levels of *knowing* and *comprehending* if OoIs are described and side facts are presented. Further, to the level, *applying* as an abstract OL model is applied and an example is presented *synthesizing* as a common level of understanding is worked out, *analyzing* and *evaluating* as common analysis criteria are issued, and the OL concepts are regarded with respect to the CoNM.

2.4.3.1 Single-Loop Learning

Single-Loop learning (SLL) is a learning concept that issues questions about how and when decisions and actions are adapted with the aid of learning processes so that they improve a certain outcome. Within the context of learning organizations, it issues how and when organizations can increase efficiency within predefined environmental conditions.

Description: Fig. 2.46 below visualizes the Single-Loop learning concept.

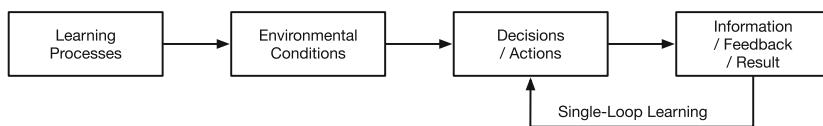


Figure 2.46 Single-Loop Learning Model for Learning Organizations (synthesized from literature)

When faced with circumstances of the real world, which are interpreted as a problematic initial situation, current outcomes are improved by the implementation of *actions* or *decisions*. So, organizational members faced with internal or external changes modify their action strategy if an output does not match the expected results (Rhodes, 1998; Becker, 2002).

Through a repetitive implementation of the same set of actions under the same *environmental conditions* and the same *learning processes*, the implementation of that set of actions is improved, which is referred to as Single-Loop learning. So, the efficiency of the current set of actions improves with every iteration and a learning cycle is started until outcomes satisfy certain outcome criteria. From the perspective of learning organizations, the SLL focuses on doing things right.

Side facts: SLL approaches in organizational contexts were first provided by Argyris and Schön (1978), who built on definitions by Ashby (1960). Their focus lies on dealing with humans and organizations with failures or imperfect results. As one of its key concepts, SLL is part of a theory of learning. Since then, various modifications have evolved (Koornneef, 2000; Koornneef and Hale, 2004).

Example: SLL can be exemplified by a manufacturing process which follows a certain process model. As a machine regularly breaks down and pauses the whole production process, the decision-making rule called “repairing machine” is implemented. This makes the production run again so that a short-term solution is realized. Through the repetitive SLL cycle, the implementation of the machine repairing is improved with every attempt of reparation and is referred to as one iteration. Since the causes for the machine’s breakdown are not covered and only its symptoms are covered, the solution produced is unsustainable.

Critical appraisal: While being a generic optimization concept, which is carried out on behalf of SLL, generally, all process and simulation modeling objects can be affected. As they are not prepared for being considered in SLL iterations, modeling language criteria from the business and process domain, the knowledge management domain, and the simulation domain can be extended under high efforts. SLL lays a particular focus on the business process context, such that corresponding modeling language criteria show better evaluations. This is mainly because of the systematic reviewing of as-is and to-be matches.

Modeling language criteria from the ANN domain cannot be met at all since SLL does not consider neuronal mechanisms in learning organizations as of yet. Indeed, neuronal modeling objects and mechanisms are not mentioned in any SLL approach. So, the corresponding ANN tool criteria are not met either.

Although SLL effects can be reflected in process and simulation models, they are not prepared for corresponding tool implementations. So, the tool criteria for modeling and simulations tools are not met in any case.

Since the SLL generally enables the optimization of current models so that a certain criteria improves, it has potential to be a foundation for the CoNM. This

particularly considers the efficient design of an underlying model by, first, the redesign under higher knowledge base, second, the redesign in iterations, and third, the redesign with regard to a key criteria. This further enables the application of SLL in ANN contexts so that ANN mechanisms are improved and the CoNM is enabled indirectly.

2.4.3.2 Double-Loop Learning

Double-Loop Learning (DLL) is a learning concept that issues questions about how and when environmental conditions are adapted with the aid of learning processes. This includes underlying beliefs and assumptions so that a certain outcome improves. Within the context of learning organizations, it issues how and when organizations can increase the set of potential actions, which further improve a certain outcome.

Description: Fig. 2.47 below visualizes the Double-Loop learning concept.

When faced with circumstances of the real world, which are interpreted as problematic initial situation, current outcomes are improved by the implementation of *actions* or *decisions*. Since *environmental conditions* are reviewed and possibly adapted before a current set of actions is implemented, further potential actions are identified. This includes mental models and patterns such as underlying beliefs and assumptions, individual's motives, and informal and ingrained practices, among others. So, organizational members faced with internal or external change modify their expectations about objectives if an output does not match the expected results and prerequisites for SLL are established (Rhodes, 1998; Becker, 2002).

Through a repetitive questioning of an attractive set of actions under the same *learning processes*, the identification of the best set of actions is improved, which is referred to as Double-Loop learning. So, the choice of attractive actions improves with every iteration and a learning cycle is started until outcomes satisfy certain outcome criteria. From the perspective of learning organizations, DLL focuses on doing the right things.

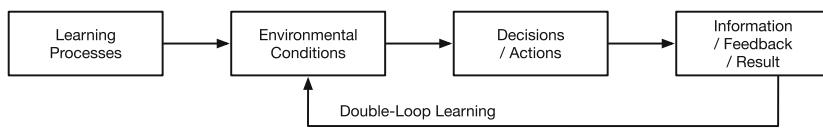


Figure 2.47 Double-Loop Learning Model for Learning Organizations (synthesized from literature)

Side facts: DLL approaches in organizational contexts were first provided by Argyris and Schön (1978), who built on definitions by Ashby (1960). Here, the focus lies on the dealing with humans and organizations with failures or imperfect results. As one of its key concepts, DLL is part of a theory of learning. Since then, various modifications have evolved (Koornneef, 2000; Koornneef and Hale, 2004).

Example: DLL can be exemplified by a manufacturing process which follows a certain process model. As a machine regularly breaks down and pauses the whole production process, decision-making rules addressed by a SLL do not sustainably improve the initial situation. For instance, the decision-making rule called “repairing machine” makes the production run again, but it does not eliminate causes for a machine’s overheating. So, the adaptation of the environmental conditions, the consideration of cooling processes within the process model, will focus on causes and create a sustainable solution. Through the repetitive DLL cycle, the identification of best environmental conditions is improved with every iteration.

Critical appraisal: As a generic optimization concept, DLL generally is able to affect all processes and simulation modeling objects. So, modeling language criteria from the business and process domain, the knowledge management domain, and the simulation domain can be extended under high efforts. Only elements from the business process context show better evaluations for corresponding modeling language criteria since as-is and to-be matches are part of DLL. Accordingly, the particular question of current goals is considered.

Since the DLL does not consider neuronal mechanisms in learning organizations as of yet, modeling language criteria from the ANN domain are not met and the corresponding ANN tool criteria cannot be satisfied. Further, modeling and simulation tool criteria cannot be met because the adaptation of process and simulation models has not been prepared till date.

Since DLL generally questions current optimization attempts of models so that a certain criteria improves, it can potentially be part of the foundation for the CoNM. This particularly considers the effective design of an underlying model by, first, the fundamental redesign under better knowledge base, second, the fundamental redesign in iterations, and third, the fundamental redesign with regard to a key criteria. This further enables the application of DLL in ANN contexts such that ANN mechanisms are improved and the CoNM is enabled indirectly.

2.4.3.3 Deutero-Learning

Deutero-Learning is a learning concept, that issues questions about how and when learning is carried out, so that habits and mental patterns are affected (Lutterer,

2012). Within the context of learning organizations, it issues how and when organizations learn (Schön, 1975; Argyris and Schön, 1978; Preskill and Torres, 1998) and so clarifies how to carry out single-loop learning (section 2.4.3.1) and double-loop learning (section 2.4.3.2).

Description: Fig. 2.48 visualizes the Deutero-learning concept.

While facing circumstances of the real world, which are interpreted as problematic initial situation, current outcomes shall be improved by the implementation of *actions* or *decisions*. As the current learning processes are analyzed and questioned, current *learning processes* are improved. This includes the consideration of supporting and hindering determinants and phases (Becker, 2002).

Through a repetitive implementation, both the identification of the most attractive set of actions (DLL) and the implementation of the best set of actions (SLL) is improved, which is referred to as Deutero-learning. From the perspective of learning organizations, Deutero-learning focuses on doing the right things right, most efficiently.

Side facts: The term “Deutero-learning” was originally coined by Gregory Bateson and evolved from 1942 to 1972 in several stages (Visser, 2003; Lutterer, 2012). The final formulation refers to a framework of four different kinds of learnings: *zero-learning*, *proto-learning*, *deutero-learning*, and *trito-learning* (Bateson, 1972, p. 293) and was introduced in organization science (Schön, 1975; Argyris and Schön, 1978; Argyris and Schön, 1996; Argyris, 2003). Since then, various interpretations and specifications have evolved its understanding (Visser, 2007).

Example: Deutero-learning can be exemplified by a manufacturing process which follows a certain process model. As a machine regularly breaks down because of overheating and pauses the whole production process, the initial situation is improved by both SLL and DLL. When this happens, decision-making rules

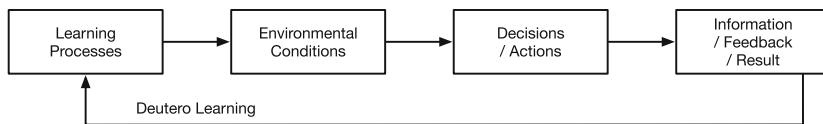


Figure 2.48 Deutero-Learning Model for Learning Organizations (synthesized from literature)

addressed by a SLL improve the initial situation so that the efficiency of a current setting is particularly increased. As mental models are addressed by a DLL, environmental conditions are adjusted so that the effectiveness of a current setting is particularly increased. Since some situations demand short-term optimizations and regard the efficiency, and others demand long-term optimizations and regard the effectiveness, an adaptation of learning processes supports efforts focusing on the best learning forms.

Through the repetitive Deutero-learning cycle, the consideration of the most suitable forms of learnings are improved with every iteration. This refers to the identification of the best environmental conditions and most efficient ways to carry them out, which is again improved with every iteration.

Critical appraisal: Deutero-learning enables, as a generic optimization concept, the optimal affecting of any process and simulation modeling object. So, modeling language criteria from the business and process domain, the knowledge management domain, and the simulation domain can be satisfied and demand high implementation efforts. Elements from the business process context show even better evaluations for corresponding modeling language criteria because of the explicit dealing of as-is and to-be matches.

Deutero-learning does not consider neuronal modeling objects and neuronal mechanisms in any case. So, modeling language criteria from the ANN domain are not met and the corresponding ANN tool criteria cannot be satisfied. Further, the adaptation of process and simulation models is not prepared by any Deutero-learning approach. Therefore, the modeling and simulation tool criteria are not met.

Since Deutero-learning generally questions current learning attempts in organizations, which can be measured by various criteria, it can be part of the foundation for the CoNM. This particularly considers the effective design of learning mechanisms by, first, the SLL realization in iterations, second, the DLL realization in iterations, and third, the learning with regard to a key criteria. This further enables the application of Deutero-learning in ANN contexts, such that ANN mechanisms are improved and the CoNM is enabled indirectly.

2.4.3.4 Critical Appraisal of Organizational Learning

Faced with the aforementioned organizational learning approaches and its interim conclusions, it seems that today's learning approaches are able to consider numerous aspects optimized in learning iterations so that a certain criteria is optimized, but they are still immature and do not consider ANN techniques. This is further elaborated upon in the following:

- None of presented organizational learning approaches considers neuronal modeling objects. Therefore, an integration of ANN objects with currently available organizational learning concepts is missing. Further, an integration of organizational learning objects with currently available ANN-based learning concepts is also missing.
- None of presented organizational learning approaches considers neuronal processes. Therefore, an integration of ANN mechanisms with currently available organizational learning approaches is missing. Further, an integration of organizational learning mechanisms with currently available ANN-based learning approaches is also missing.

Therefore, a research gap becomes visible here.

A synthesis of any organizational learning approach presented here, which will be enriched with ANN modeling objects and required concepts and mechanisms, will support a broad acceptance of the CoNM in corresponding research communities. Further, a synthesis ensures its broad application possibilities. A synthesis must assure that complex models do not reduce the ease of use for any realistic setting, instead of constructing impractical modeling approaches (Curtis, Kellner, and Over, 1992).

Hence, for the construction of a CoNM, a compromise must be identified between the holism of modeling items required and their practicability. In principle, a practical and holistic synthesis must address the following:

- It must look broadly for organizational learning loops regarding efficiency (section 2.4.3.1) and provide modeling objects for its focus.
- It must look broadly for organizational learning loops regarding effectiveness (section 2.4.3.2) and provide modeling objects for its focus.
- It must look broadly for organizational learning loops regarding learning behavior (section 2.4.3.3) and provide modeling objects for its focus.
- ANN techniques must be harmonized with the understanding of organizational learning. This includes process modeling, simulation modeling, and knowledge modeling as individual learning domains.

The creation of a CoNM will, therefore, consider those points and it will be based on a practical synthesis of organizational learning approaches presented here.

2.4.4 Conclusion About POPs and Organizational Learning

The conclusion about of process optimization is based on the following. First, process optimization philosophies (POPs) are appraised. Then, organizational learning approaches (OLs) follow and a process optimization-specific facet is then identified.

Optimization Philosophies: All approaches to optimize processes, so-called process optimization philosophies, are based on a systematic way to consider knowledge in order to carry out changes using current models. The consideration of neuronal perspectives, neuronal modeling objects, and an integration with components from a simulation modeling domain have not been identified in any approach (see each individual interim conclusion). Since POPs were defined in order to carry out improvements for process models, which do not refer to improvements coming from a CoNM, an appropriate CoNM optimization concept is still missing. Hence, to date, POPs have neither been evaluated to be the foundation for the CoNM (as building blocks on various levels), nor have POPs been evaluated to evaluate elements required by an CoNM.

In order to decide whether current POPs can at least partly identify a CoNM foundation or which criteria is suited to be part of a CoNM criteria catalog, the following appraises criteria category-wise:

1. Since the CoNM is intended to be carried out in a holistic and general setting, any kind of *knowledge* considered by POPs is relevant since these focus on aspects which are accepted in the process optimization community. This can refer to one-time knowledge considered in BPR attempts, incremental knowledge use considered in CIP attempts, and knowledge in the form of standards and best practices. Criteria trying to apply accepted concepts from the process domain in the ANN process domain are highly relevant.
2. Since the CoNM is intended to be carried out in a holistic and general setting, any *phase* and *activity* provided by POPs is relevant since phases and corresponding activities represent proceeding options which are accepted in the process optimization community. Criteria trying to apply accepted concepts from the process domain in the ANN process domain are highly relevant.

Since no single approach issued the use of neuronal techniques and concepts, which is an essential part of the ANN process domain, none of the provided POPs are able to partly stand as a CoNM foundation. Further, this means that the process optimization domain has not been brought together with AI techniques till date. A research gap becomes visible here, which asks for the following: first, the construction of a CoNM

because anything similar is missing; second, a CoNM POP selection system that extends common criteria and is able to evaluate current POPs; and third, an analysis of currently available POPs for the use of CoNM relevant aspects, such as neuronal techniques, specific modeling objects, and common tool criteria.

Organizational Learning Approaches: Most OLs are based on individual *loops* with special *focus*, each having a unique optimization strength. The consideration of neuronal perspectives, neuronal modeling objects, and an integration with components from the OL domain have not been satisfactorily identified in any approach (see each individual critical appraisal). Since OLs were defined in order to carry out organizational improvements, which do not refer to improvements coming from a CoNM, an appropriate CoNM OL is still missing. Hence, OLs have neither been evaluated to be the foundation for the CoNM (by providing commonly accepted learning loops), nor have OLs been evaluated to be part of a CoNM till date.

In order to decide whether current OLs can at least partly be used as a CoNM foundation or which learning loop and focus is suited to be part of a CoNM evaluation catalog, the following appraises aspects category-wise:

1. Since the CoNM is intended to be carried out in a holistic and general setting, any *modeling item* provided by OLs is relevant since items focus on aspects which are accepted in the simulation modeling community. Here, one can find *learning processes*, *environmental conditions*, *decisions*, *actions*, *information*, *feedback*, and *results*. Criteria trying to apply accepted concepts from the simulation domain in the ANN process domain are highly relevant.
2. Since the CoNM is intended to be carried out in a holistic and general setting, any *loop* provided by OLs is relevant since each establishes a learning and optimization focus which is accepted in the process optimization community. Criteria trying to apply accepted concepts from the process optimization domain in the ANN process domain are highly relevant.

Since no single OL satisfactorily provided items or loops regarding the use of neuronal techniques and concepts, which is an essential part of the ANN process domain, none of provided OLs are able to solely serve as a CoNM simulation modeling language. Further, this means that the process optimization domain has not been brought together with AI techniques till date. A research gap becomes visible here, which asks for the following: first, the construction of a CoNM OL approach because anything similar is missing; second, a CoNM OL selection system that extends common criteria and is able to evaluate current OLs to be part of a CoNM foundation; and

third, an analysis of currently available OLs for the use of CoNM relevant aspects such as neuronal techniques, specific modeling objects, and common tool criteria.

Facit: Neither process optimization philosophies nor OLs consider the integration of the process optimization domain and the ANN domain; POPs and OLs do not provide ANN and Deep Learning criteria. Therefore, a research gap within the process domain becomes visible here. This asks for the following: first, the construction of a CoNM taxonomy focusing on POPs and OLs; second, the design and demonstration of a CoNM selection and evaluation framework considering POPs and OLs as objects of investigation; and third, the implementation and demonstration of a CoNM optimization.

2.5 Deep Learning with Neuronal Networks

While process modeling focuses on the structural and behavioral modeling of processes (section 2.1), knowledge modeling brings in the context of knowledge by the provision of structural knowledge elements as well as the consideration of their dynamic effects (section 2.2), and simulation modeling enables the controlled dealing of different kinds of systems in realistic imitations (section 2.3). Thus, deep learning (DL) attempts to bring in the context of self optimizing systems that provide structural knowledge as well as realize a dynamic behavior. Since ANN learning processes can even be interpreted as adaptations of processual structures, ANN knowledge must be considered to be structural as well as processual and the terms *knowledge* and *knowing* are equivalent on the neuronal level (compared to Neuweg, 1999, p. 135; Renzl, 2004, p. 38; Gronau, 2009, p. 6 in ANN contexts).

Ideally, common approaches to represent processes are combined with knowledge modeling approaches. Jointly, they are mapped to ANN modeling approaches so that the behavioral side of knowledge can be recognized from the processual perspective of ANN contexts as well. Here, a potential can be identified in the recognition of knowledge within the processual behavior of complex ANN structures. The underlying hypothesis of this contribution is that insights in the working of ANN can be realized on behalf of this integration and lead to, first, their function as a sub-simulation system, second, their representation of complex process participants, third, their representation of processes, fourth, their knowledge generation engine, and fifth, the optimization of processes because of the adaptation of ANN.

The Deep Learning context is characterized by the following. First, basic concepts and definitions are established. These range from the initial biological motivation (section 2.5.1), that can be intended to be transferred to economic contexts

and further address the dealing with artificial neuronal networks such as the specification for learning tasks (section 2.5.2), their access on behalf of a codification (section 2.5.3), and the design of learning problems (section 2.5.4). Basic ANN techniques are introduced in section 2.5.5 such that on-building learning procedures can be specified on behalf of them (section 2.5.6). Then, state-of-the-art approaches for establishing an interpretation of ANN compartments are issued in section 2.5.7. Presented jointly with common network archetypes in section 2.5.8, these will stand as archetypes for the economic reuse of the neuronal modeling conceptualized by this research. Second, different kinds of modeling approaches for ANN are provided based on a literature review. Here, relevant perspectives are identified that will be the foundation for the creation of the *ANN Process Taxonomy* (section 4.1.1) and stand as a standard for process modeling languages. So, in section 2.5.9, a domain-specific research need is identified by the examination of prominent notations used to represent ANN models.

2.5.1 Biological Motivation

Orienting to biological phenomena, the CoNM intends to make the working behavior of biological systems, more precisely the human brain, accessible for non-biological systems such as process and business process modeling domains and economic contexts. With this, it tries to function as a bridge between both domains. Therefore, the CoNM follows *bionic* attempts to technically realize insights about biological systems and tries to consider the process understanding of organizations as well as the world on the basis of evolution over centuries of optimized systems.

Mostly, the behavior of a biological brain is modeled by mathematical algorithms. These are faced with learning tasks so that an artificial version of the human brain is enabled to solve difficult problems. Attempts to mathematically represent brain functionality first started in 1940 (McCulloch and Pitts, 1943). To date, they have evolved in the *neuroscience* discipline and trends such as *deep learning* (Schwaiger and Steinwendner, 2019, p. 28).

2.5.1.1 The Nervous System

Neuronal signal processing is carried out within the *nervous system*. This consists of the *central nervous system*, which includes the *brain* and *spinal cord*, various *sensory systems* collecting information throughout the body, and the *motor system* which controls movements (Schwaiger and Steinwendner, 2019, p. 242). These are made out of neuronal cells and non-neuronal cells or the so-called *glia cells*, providing structural support and protection to the neuronal cells (Jessen K.R, 1980). Further,

they support the construction of connections among neurons, nourish neurons, dispose of dead cells, and recycle old neurotransmitters (Schwaiger and Steinwendner, 2019, p. 247). Although most of the signal processing is realized by the brain, a pre-processing or post-processing by the design of body compartments, such as the retina of the eye, are significant for signal processing (Nauck et al., 2013, p. 5).

Within the nervous system, information is constituted by changes in two ways: first, an *electrical potential*, and second, the number of nerve impulses per second or the so-called *fire rate* (Nauck et al., 2013, p. 7). Realistically, more than 100 impulses can occur per second (Kruse et al., 2015, p. 11), which are processed mainly by neuron interaction.

Neurons communicate with each other by transmitting signals either by chemical or electrical mechanisms (Schwaiger and Steinwendner, 2019, p. 30). Hence, neurons are suited to be routing points for incoming and outgoing signals. Compared to the signal processing of computer systems, they operate in a massive parallel manner and are very robust against failures. Further, they are robust with regard to breakdowns since they can build redundant structures and they can reorganize during the course of learning.

Since the operation of the nervous system, particularly the human brain, is resource intensive, its entire management and its structural and behavioral organization aims for economic resource handling.

2.5.1.2 The Brain

With an average weight of 1400g, the human brain consists of about 86 billions nerve cells also referred to as *neurons*, 10^5 synapses and about 1,000 and up to 10,000 connections of any neuron (Nauck et al., 2013, p. 7; Deru and Ndiaye, 2019, p. 58). Jointly, these structures are particular suited for the perception of visual (seeing), auditory (hearing), olfactory (smelling), gustatory (tasting), and tentative (touching) sensory input and the generation of mental and physical reactions so that complex movements, creative solutions, and other such activities can be carried out efficiently and promptly (Schwaiger and Steinwendner, 2019, p. 29). Although the natural switching time of a biological neuron was determined to be 1 millisecond (Lämmel and Cleve, 2012, p. 190), it is considered slow when faced with the cognitive processing of complex tasks, such as the recognition of objects, and the parallel working of the nervous system which allows quick processing.

The youngest part of the brain that has developed in evolutionary terms, which leads to the intelligent performance of the brain, is the *neocortex* (Ritter, Martinetz, and Schulten, 1991, p. 17). By percentage, the neocortex is the biggest part of the brain. It obtains a surface of about 0.2 square meters, is extensively folded, and has a thickness of about 2–3 millimeters (Ritter, Martinetz, and Schulten, 1991, p. 17). In

this part of the brain, regions can be identified which establish a specialization for the certain tasks performed. For example, there are fields specialized in visual perception (visual cortex), motion control (motor cortex), tasting (somatosensory cortex), or the linking of information coming from different sensory systems (association fields) (Ritter, Martinetz, and Schulten, 1991, p. 17).

2.5.1.3 The Neuron

Building on the biological neuron schema (Rougier, 2019), Fig. 2.49 presents the neuron's compartments which are essential for signal processing.

For a neuron, incoming signals refer to chemical signals that mostly arrive at the neuron by widely ramified *dendrites*. Here, rather small distances of 400 micrometers and slow transfer times are realizable (Ritter, Martinetz, and Schulten, 1991, p. 19). Having been processed by the neuron itself, the delivery of signals is realized by *axons* that transport electric signals quickly over wide distances of up to one meter at 0.5–130 meters per second (Nauck et al., 2013, p. 7). As the axon is covered by several *myelin sheaths*, the electric signal is protected by the axon's surroundings. *Nodes of Ranvier* strengthen the signal during the course of transmission. Commonly, these are 1 micrometer in size and occur along the axon every 0.2–2 millimeters (Schmidt et al., 2013).

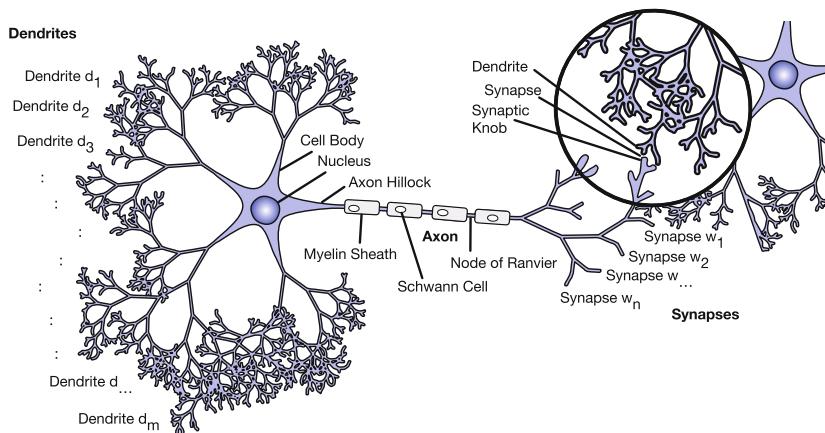


Figure 2.49 Schematic Drawing of Compartments of a Neuron (synthesized from literature and created to show the consistency of neuronal networks)

Most of the neurons only provide one axon that transmits the signal to various neurons via a tree-like ending structure (Schwaiger and Steinwendner, 2019, p. 29). Each ending of an axon (*synaptic knob*) docks to the cell body of a neuron or its dendrites (Ritter, Martinetz, and Schulten, 1991, p. 19). The very small gap between both compartments is referred to as a *synapse*. Typically, its distance is not more than 10–50 nanometers (Kruse et al., 2015, p. 10). Synapses function as an interface so that the chemical signal, which occurs between the neurons, is transformed into an electrical signal for transmission within the neuron.

2.5.1.4 The Signal Transmission

The signal transmission from neuron to neuron is realized by two kinds of signal transmissions. While the chemical transmission provides failure tolerant characteristics over short distances and affects many dendrites, the electrical signal transmission realizes an efficient and fast signal transmission over wide distances. The biological mechanisms are exemplified in the following sections.

Chemical Signal Transmission: Commonly, the inside of the cell body provides a negative potential which refers to about 70 millivolts (Kruse et al., 2015, p. 10). This is because of the concentration of negative ions inside the cell body and more positive ions outside the neuron. This results from continuously working *ion pumps*. As a signal is provided by an axon, *neurotransmitters* are unleashed, which affect the electrical potential of the target dendrite. To date, more than 200 different kinds of neurotransmitters have been identified (Xu and Xue, 2019; IUPHAR and BPS, 2019), each resulting in potential changes with different sizes and direction. As neurotransmitters are collected by *receptors* having a specialization of a particular kind of neurotransmitter, the chemical signal transferred is either *inhibiting* or *exciting* (Schwaiger and Steinwendner, 2019, p. 241).

Being connected to many axons, the cell body is suited as a central collection point for potential changes. If the total potential change is big enough, the *axon hillock* will be *depolarized* because of the intrusion of positive *sodium ions*. As positive ions, such as *potassium ions*, are extruding then, in about 1 millisecond Kruse et al., 2015, p. 11, the negative potential is rebuilt quickly. This short-term potential change probably does not last for more than 1 millisecond Kruse et al., 2015, p. 11 and is sufficient to start an electric signal transmission. Finally, the original concentration of ions is rebuilt by ionic pumps so that the original negative potential is achieved.

Electrical Signal Transmission: Because of the depolarization of the cell body during the course of potential change, the axon hillock digitizes the signal and starts

the transmission of an electric impulse which is referred to as *action potential*. This runs through the axon by jumping from a node of Ranvier to the next because of the following. Along the axon, in between the myelin sheets, a great density of voltage-controlled ion channels can be found. Once these are affected by the arriving action potential, they start to strongly intrude positive ions, such as sodium ions or potassium ions, and the current node of Ranvier depolarizes (Ranvier, 1872).

Because of the excess of positive ions, an electrical field establishes that makes negative ions, such as *chlorine ions*, from the neighboring node of Ranvier move upwards the current node and positive ions move downwards the current node of Ranvier (Schwaiger and Steinwendner, 2019, p. 247). If this charge displacement exceeds a certain threshold sufficiently, the neighboring node of Ranvier starts depolarizing as well. The signal transmission thus described is referred to as *salatory conduction*.

Neuronal Information Processing: So far, various terms regarding the information processing within the brain have been widely discussed. Some suggest different kinds of cognitive energetic pools associated with information processing so that one can find a pool for the *effort*, the *arousal*, and the *activation* (Sergeant, 2000; Sergeant, 2005). Similarly, Halperin and Schulz (2006) suggest the pools of *arousal* and *alertness* and Sikström and Söderlund (2007) suggest different levels of dopamine resulting in cortical *hypoorousal*. Based on experimental studies, Barry et al. consider the frequency of electrical activity of the brain to be a robust indicator of arousal across development (Barry et al., 2004; Barry et al., 2007). Thus, the arousal can be defined as the “current energetic level of the organism” and activation as a “separable tonic measure reflecting the task-related mobilization of energy needed to perform a task” (VaezMousavi et al., 2007).

Since the theoretical separation of effort, arousal, activation, and alertness have been discussed, and these terms are often used interchangeably (Loo et al., 2009), this contribution focuses on the following: first, the dealing with neuronal activity in general as it is the most basic level of information processing; and second, the undifferentiated regard of effort, arousal, activation, and alertness as simplification modeling different kinds of biological and chemical processes.

2.5.1.5 Further Principles

Even though ANNs intend to reproduce the working behavior of the human brain (Lämmel and Cleve, 2012, p. 190) and the brain of animals (Kruse et al., 2015, p. 9), they strongly simplify biological mechanisms so that they hardly have to do with biological models. The following sections collect joint principles.

Lateral Inhibition: Based on the appearance of neurons, the cerebral cortex provides two kinds of neurons: First, the *pyramid cells* mostly having one long axon with split endings being connected excitatorily; and second, the *stellate cells* mostly having several short axons being connected inhibitarily. So, an architecture is built that stores relevant information over wide distances using pyramid cells and considers stellar cells for regional stabilization by inhibiting stellate cells (Ritter, Martinetz, and Schulten, 1991, p. 20).

Microcolumns: The cerebral cortex shows many regions in which groups of neighboring neurons cooperate so that these function as entities realize higher functional tasks (Mountcastle, 1978; Blasdel and Salama, 1986). Commonly, these are grouped into cylinders taking up room of about one tenth of a millimeter in diameter and being referred to as *microcolumns* (Ritter, Martinetz, and Schulten, 1991, p. 20). For example, one of those cylinders might be responsible for the analysis of a certain stimulus such as an edge detection of visual stimuli or the activation of a muscle. Because of the lateral inhibition, neurons cannot strictly be associated to only one microcolumn. Hence, neighboring microcolumns cannot be differentiated clearly and rather, they provide neurons with a gradual sense of belonging.

Cortical Fields: On a higher organizational level, microcolumns are arranged to specialized fields, each being responsible for a particular subtask and known as a *cortical field*. To date, more than 80 cortical fields are known in the human brain (Ritter, Martinetz, and Schulten, 1991, p. 20). These refer to tasks employed for the identification of edges, colors, velocities, speech, faces, spatial orientation, planning and realization of movements, etc.

Cortical Maps: Because of the task specialization of cortical fields, one can often identify the building of maps (Ritter, Martinetz, and Schulten, 1991, p. 20). If one is able to identify simple attributes for the task a cortical field is responsible for, a systematic variation of these attributes will be positioned along the two spatial dimensions of the field's surface. An example can be found in the acoustic signal processing of owls. This shows attributes as a two-dimensional map of the azimuth and height of the signal direction (Knudsen, Lac, and Esterly, 1987). Another example refers to the somatosensory cortex of apes that shows attributes following the presence of body compartments (Kaas et al., 1979). So, regions focusing on the signal processing coming from arms can be separated by regions focusing on signals coming from fingers, lips, legs, and so on.

Most of the cortical maps can be categorized by one of the following groups (Ritter, Martinetz, and Schulten, 1991, p. 21): first, primary and secondary *sensory*

fields, focusing on the preprocessing and directly fed by sensory receptors; second, *association fields*, processing preprocessed sensory signals; and third, primary and secondary *motor fields*, focusing on the activation of muscles.

Layers: As any cortical field and cortical map is connected with other cortical fields and cortical maps as well as with non-cortical brain structures and nerve system structures, an extremely coupled and parallel working system is constructed. Despite this complexity, the cortex follows a surprisingly homogeneous structure with six levels (Ritter, Martinetz, and Schulten, 1991, p. 21). Level I focuses on the main task of the cortical field and cortical map. Level II and III provide connections to neighboring regions and support association fields. Layer IV functions as an input layer for the cortical field and cortical map. Here, further connections to foreign regions are routed. Level V and VI send recurrent connections to neurons that control level IV.

Topographic Connections: Connections among modules, independent of micro-columns, cortical fields, cortical maps, etc., follow one organization principle: neighboring neurons of a source module tend to be connected with neighboring neurons of an target module (Ritter, Martinetz, and Schulten, 1991, p. 22). So, although the wiring is realized through various modules, the stimulus perception of a sensory surface is mapped to the responsible module and a topographical organization in the sense of a map is realized.

Interim Conclusion: Beside the consideration of biological compartments and mechanisms presented in this section, the consideration of biological principles presented is required for the construction of a CoNM, such that organizations can be organized in analogy of, for instance, nervous systems, brains, and neocortex.

2.5.2 Learning Tasks

The advantage of ANNs is that they can be trained with the aid of example data, so that they perform a predefined learning task on unseen data well. Mostly, the learning procedure of ANN goes along with parameter adjustments such as connection weights and threshold values, among others so that a certain criteria is optimized. Depending on the kind of training data and the criteria to be optimized, the *learning task* can be differentiated and the corresponding techniques (or learning modes) and kinds of ANN architectures can be selected (Kruse et al., 2015, p. 40, 41; Deru and Ndiaye, 2019, p. 34; as well as Schwaiger and Steinwendner, 2019, p. 313).

In accordance with Deru and Ndiaye (2019), the following presents common learning tasks individually. While the first sub-section deals with labeled learning material (supervised learning), the second sub-section focuses on a learning with unlabeled material (unsupervised learning). The third sub-section characterizes a learning on behalf of a reinforcement (reinforcement learning) so that the major learning tasks are established.

As there are further learning modes dealing with both labeled and unlabeled learning material (semi-supervised learning) and enabling an interactive version of semi-supervised learning (active learning) (Deru and Ndiaye, 2019, p. 36, 38), the following only focuses on basic learning modes.

2.5.2.1 Fixed Learning Task

As a system, an ANN is intended to realize a specifiable learning outcome because one knows the answer for every input of the training material, but one can identify a fixed learning task (Nauck et al., 2013, p. 36; Kruse et al., 2015, p. 40):

Definition 21 (Fixed Learning Task).

A fixed learning task for an ANN having n input neurons, which means the set $U_{in} = \{u_1, \dots, u_n\}$, and m output neurons, which means the set $U_{out} = \{u_1, \dots, u_m\}$, refers to a set of r patterns $L_{fixed} = \{l_1, \dots, l_r\}$ providing the pattern $l = (\underline{\mathbf{i}}^{(l)}, \underline{\mathbf{o}}^{(l)})$, such that the ANN's task is to perceive the externally given input vector $\underline{\mathbf{i}}^{(l)} = (ext_{u_1}^{(l)}, \dots, ext_{u_n}^{(l)})$ at its input neurons n and produce the corresponding output vector $\underline{\mathbf{o}}^{(l)} = (ext_{v_1}^{(l)}, \dots, ext_{v_m}^{(l)})$ at its output neurons m with the aid of arbitrary complex neuronal structures and mechanisms between input and output neurons.

Since fixed learning tasks provide an input and output relation, the derivation of an error is suited to act as optimization criteria. The better the network is trained, the smaller a certain error will be. Following this definition, an ANN is able to approximate functions as the input values can be generated and corresponding output values can be derived easily. Further, they can categorize any sensory input to arbitrary complex classes if the corresponding classes are prepared. They are able to predict the future if a sensory perception of one time step t is considered as input and not necessarily the same sensory perception of the consecutive time step $t + 1$ is considered as output. Training in the context of a fixed learning task is referred to as *supervised learning* because of the metaphor of supervision of a learning output.

2.5.2.2 Free Learning Task

As a system, an ANN is intended to realize a learning outcome by itself because one does not know the answer for training material input, one can identify a free learning task (Nauck et al., 2013, p. 37; Kruse et al., 2015, p. 41):

Definition 22 (Free Learning Task).

An *unfixed learning task* for an ANN having n input neurons, which means the set $U_{in} = \{u_1, \dots, u_n\}$, and m output neurons, which means the set $U_{out} = \{u_1, \dots, u_m\}$, refers to a set of r patterns $L_{free} = \{l_1, \dots, l_r\}$ providing the pattern $l = (\underline{\mathbf{i}}^{(l)}) = (ext_{u_1}^{(l)}, \dots, ext_{u_n}^{(l)})$ at its input neurons n and produce an output vector $\underline{\mathbf{o}}^{(l)} = (ext_{v_1}^{(l)}, \dots, ext_{v_m}^{(l)})$ at its output neurons m with the aid of arbitrary complex neuronal structures and mechanisms between input and output neurons.

Since free learning tasks only provide an input, the derivation of an error cannot function as optimization criteria. Here, a measurement of the similarity of input patterns is suited to function as optimization criteria. The better the network is trained, the more similar its output will be when it is faced with similar inputs. Following this definition, an ANN is able to identify groups of similar input vectors, which is called *clustering*. Further, ANNs are able to recognize patterns and complete fragmented input vectors. They are also able to prioritize given external constraints. The training in the context of a free learning task is referred to as *unsupervised learning* because of the absence of a learning output.

2.5.2.3 Reinforced Learning Tasks

As a system, an ANN is intended to realize a learning outcome by itself, and as any outcome is *rewarded* or *penalized*, one can identify a reinforced learning task:

Definition 23 (Reinforced Learning Task).

A *reinforced learning task* for an ANN having n input neurons, which means the set $U_{in} = \{u_1, \dots, u_n\}$, and m output neurons, which means the set $U_{out} = \{u_1, \dots, u_m\}$, refers to a set of r patterns $L_{reinforced} = \{l_1, \dots, l_r\}$ providing the pattern $l = (\underline{\mathbf{i}}^{(l)}, \underline{\mathbf{o}}^{(l)}, \underline{\mathbf{r}}^{(l)})$, such that the ANN's task is to perceive environmental surrounding $\underline{\mathbf{s}}^{(l)}$ as an externally given input vector $\underline{\mathbf{i}}^{(l)} = (ext_{u_1}^{(l)}, \dots, ext_{u_n}^{(l)})$ at its input neurons n and produce an output vector $\underline{\mathbf{o}}^{(l)} = (ext_{v_1}^{(l)}, \dots, ext_{v_m}^{(l)})$ at its output neurons m with the aid of arbitrary complex neuronal structures and mechanisms between input and output neurons. This represents an action $\underline{\mathbf{a}}^{(l)} = (ext_{a_1}^{(l)}, \dots, ext_{a_m}^{(l)})$ out of

set of actions $A_{reinforced} = \{a_1, \dots, a_p\}$ and tries to maximize the cumulative and discounted reward of the ANN $\underline{\mathbf{r}}^{(l)}$, which is derived by an independent interpreter who evaluates the environmental transition from $\underline{s}^{(l)}$ to $\underline{s}'^{(l)}$ because of the realization of $\underline{\mathbf{a}}^{(l)}$.

This definition builds on the understanding of reinforcement learning which is modeled as a Markov decision process (Witten, 1977; Sutton and Barto, 1998). Here, a transition refers to (s, a, s') and an immediate reward $R_a(s, s')$ after transition from s to s' through action a . The agent that is faced with the environment and agent states selects its actions with the aid of a map called *policy* π . The training in the context of a reinforced learning task is referred to as *reinforcement learning* because of the consideration of a reward.

2.5.3 Knowledge Codification and Encoding

In distinction to knowledge representation formalisms (section 2.2.1.2), each learning tasks demands for individual knowledge, information, and data *codification*, further referred to as *encoding*. While some learning tasks demand for the perception and production of real numbers (prediction, function approximation, etc.), others demand for the dealing with symbolic attributes (colors, classes, opinions, and ordinal scales, etc.). In order to enable ANN to deal with symbolic attributes on behalf of a codification scheme, different knowledge forms can be transformed into machine readable, practical, understandable, or effective forms which might not be interpretable by humans at once.

Encoding: While some input signals, such as simple data streams from production machines, can be accessed by ANN directly or do not demand for more than a simple transformation, more sophisticated and on-building forms, such as information and knowledge, can require particularly creative codification solutions. In order to enable ANN to deal with symbolic attributes, the following codification strategies can be applied (Kruse et al., 2015, p. 42):

First, symbolic attributes can be transformed to numbers. As this implies attributes to have a certain inner distance, this is not adequate.

Second, symbolic attributes can be numbered consecutively. As this implies attributes to have an ordering and a certain inner distance, this is not adequate either.

Third, each symbolic or nominal value is mapped to an individual input neuron (for the perception as input codification) or output neuron (for the production as

output decodification). Since each neuron is in accordance with an attribute value, its incoming or outgoing signal is set to 1 if it is faced with the attribute it is representing and it is set to 0 for other attributes. This is referred to as *1-out-of-n codification*.

Fourth, each symbolic or nominal value is mapped to an individual input neuron (for the perception) or output neuron (for the production) with the exception of one attribute. The one exceptional attribute is codified or decoded by all neurons being set to 1. Since this allows the dealing of n attribute values with $n - 1$ neurons, this is referred to as *1-out-of-(n-1) codification*.

Decoding: While some output signals, such as simple data streams being used in third party systems, can be generated by ANN directly or do not demand for more than a simple transformation, more sophisticated and on-building forms of information and knowledge can require particularly creative decoding solutions.

If knowledge, information, and data have been codified for learning tasks, one has to pay attention for the interpretation of signals during the inner processing of the ANN and the results being produced by the ANN (Luong et al., 2015). Before encoded signals can be interpreted by humans, they might need to be decoded and the following speaks for *anti codification* or for *decoding*. Here, this is essential if the signal processing has been affected by codification invariant changes. If the codification base of an input signal has not been affected, a decoding can be easily realized. Otherwise, a common codification base needs to be identified before decoding starts.

Trading-off Codification and Decodification: While one needs to satisfy conditions of a learning tasks a codification scheme requires, the design of a learning task can creatively simplify the learning effort required by an ANN to solve a learning task. The more transformation, decoding, and encoding efforts the learning tasks includes, the more the ANN will be challenged. Here, simple tasks can be realized by non-AI components. This further increases the probability of successfully realizing a learning task. As some codification schemes simplify the learning effort, a practical compromise in codification and anti codification in the design of a learning tasks is to be identified.

2.5.4 Learning Problems

As each learning task is realized by following an individual learning objective or purpose, it also follows the type of a problem to be solved (Deru and Ndiaye, 2019).

Hence, in accordance with Deru and Ndiaye (2019, p. 34), the following presents common learning problems individually. Further, the first sub-section introduces clustering, the second sub-section refers to classification problems, and the third sub-section refers to regressions and prediction problems.

2.5.4.1 Cluster Problems

The way to grouping administered items so that items of the same group resemble one another more than these items would resemble to those in other groups, is considered as cluster problem. In contradiction to a classification problem (section 2.5.4.2), here, the intention refers to the identification of groups within a set of data.

Originally defined by Driver and Kroeber (1932), Zubin (1938), Tryon (1939) as well as Cattell (1943), the analysis of clusters has lead to different understandings of what constitutes a cluster and how to efficiently find them (Chan et al., 2016). So, one can consider small distances between cluster members, intervals, statistical distributions, or dense areas of the data space as the basis of assessment which characterize the *proximity*. Further, one can consider different *forms of clusters*, as these can be overlapping, flat or hierarchical, hard or soft, and so on (Hennig et al., 2015). The question of whether clusters identified are attractive in terms of a certain research intention can only be answered by the interpretation of experts. More attractive might mean that these lead to plausible insights, verify assumptions, lead to an improved performance analysis or provide adequate validation KPIs.

According to Hennig et al. (2015, p. 2), considering the context of ANN, cluster problems intend to make an ANN perform a mapping from a continuous input space \mathbb{R}^d , also called a data set D , into L disjoint subsets of D , denoted by the finite set of classes $Y = \{C_1, C_2, C_3, \dots, C_l, \dots, C_L\}$.

In the learning phase, the parameters of ANNs are determined from a finite training set S containing m data points

$$S = \{(\underline{\mathbf{x}}^m) | \underline{\mathbf{x}}^m \in \mathbb{R}^d, m = 1, \dots, M\}, \quad (2.15)$$

so that each data point $\underline{\mathbf{x}}^m$ belongs to a cluster C_l , and we can say that l is the label of $\underline{\mathbf{x}}^m$. This means that each feature vector $\underline{\mathbf{x}}^m$ is unlabeled and needs to map to its class membership C_l^m . During the *recall phase*, when the learning has finalized successfully on M examples, the ANN can be faced with unlabeled observations $\underline{\mathbf{x}} \in \mathbb{R}^d$ so that it estimates their class memberships $\{y \in Y\}$.

In regard to Hennig et al. (2015, p. 5–6), clusters differ in their relation to each other and the data set D . The following presents an overview of the ways in which they do so.

Partitioned Clusters: Approaches demanding for strictly separated clusters. Here, each data item belongs to only one cluster $C \in Y$, formalizing the condition of having partitions with $C_j \cap C_k = \emptyset$ for $j \neq k$.

Overlapping Clusters: Approaches relaxing the condition of partitioned clusters of having partitions enable data items to belong to more than to one cluster. Here, we have $C_j \cap C_k \neq \emptyset$ for $j \neq k$.

Exhaustive Clusters: Approaches demanding for soft clusters and demanding to exhaust the given set of clusters ask for $\cup_{j=1}^K C_j = D$.

Hierarchical Clusters: Having overlapping clusters that follow a sequence of partitions $\cup_{j=1}^k Y_j$, where Y_j , $j = 1, \dots, k$ are partitions with $|Y_1| > \dots |Y_k|$. Here, $|Y|$ refers to the number of elements in Y such that for $C_j \in Y_j$ and $C_l \in Y_l$ with $j < l$, either $C_j \cap C_l = C_j$ or $C_j \cap C_l = \emptyset$, and the sets of lower levels are partitions of higher level sets.

Variants refer to *full hierarchies* (Hennig et al., 2015, p. 6), *pyramids* and *weak hierarchies* (Bertrand and Janowitz, 2002, p. 6), and non-hierarchical overlapping (Shepard and Arabie, 1979).

Hard Clusters: Approaches demanding each data item belong to only one cluster $C \in Y$ and establish hard cluster borders.

Soft Clusters: Approaches relaxing the hard clustering condition of having only one and only one item belonging to one cluster $C \in Y$. They provide a *degree of membership* γ_{il} to each of the L clusters. Here, Y refers to a collection of soft assignments $\{\gamma_{il}, \underline{x}_i \in S, l = 1, \dots, L\}$, which can be interpreted to be a probability distribution over the label values.

Fuzzy Clusters: Approaches demanding for a *membership function* that assigns a value $[0, 1]$ to every item in D , indicating a degree of membership to a cluster by values.

Probabilistic Clusters: Building on fuzzy clusters and assuming that values indicating the degree of membership to a cluster sum up to 1, items are assigned to clusters that maximize membership probability.

Rough Set Clusters: Upon assuming to have pairs of *upper approximations* and *lower approximations* of sets, one can identify core sets with sure members and items not certainly being associated with a cluster.

Interim Conclusion: For the construction of a CoNM, clustering problems are attractive for the design of learning tasks that focus on the building of hierarchies of processes and activities. Further, knowledge being generated by the ANN can be clustered so that network structures become identifiable, establish a similar behavioral pattern, and, therefore, realize a comparable meaning within the network. The different forms of clusters enable the CoNM as they provide more or less sophisticated conditions for its learning task designs. They will, therefore, function as part of the technical foundation for the CoNM.

2.5.4.2 Classification Problems

Classifying administered items with the aid of classification schemes, which means the arranging of things such as the perception of an example product into groups of items known as *classes* or *clusters*, is considered a classification problem. In contradiction to a cluster problem (section 2.5.4.1), here, the intention refers to the identification of the belonging of data to groups.

According to Schwenker, Kestler, and Palm (2001, p. 440), classification problems intend to make an ANN perform a mapping from a continuous input space \mathbb{R}^d into a finite set of classes $Y = \{1, \dots, L\}$. Here, L represents the number of classes.

In the learning phase, the parameters of ANNs are determined from a finite training set

$$S = \{(\underline{\mathbf{x}}^m, y^m) | \underline{\mathbf{x}}^m \in \mathbb{R}^d, y^m \in Y, m = 1, \dots, M\}. \quad (2.16)$$

This means that each feature vector $\underline{\mathbf{x}}^m$ is labeled with its class membership y^m . During the *recall phase*, when the learning has finalized successfully on M examples, the ANN can be faced with unlabeled observations $\underline{\mathbf{x}} \in \mathbb{R}^d$ so that it estimates their class memberships $\{y \in Y\}$.

Being faced with fixed learning tasks aiming at classification, the number of classes leads to the number of output neurons, and the class memberships $\{y \in Y\}$ are encoded through a 1-of-L coding into a binary vector $\underline{z} \in \{0, 1\}^L$. Other coding schemes could be used as well. However, since other schemes are not very common in pattern recognition applications (Schwenker, Kestler, and Palm, 2001), the CoNM builds on this encoding scheme and the learning tasks realize $\mathbb{R}^d \mapsto \mathbb{R}^L$.

Interim Conclusion: For the construction of a CoNM, classification problems are attractive for the design of learning tasks that focus on the reuse of hierarchies of

processes and activities. Knowledge being generated by the ANN can be classified, so that network structures are identifiable and establish a comparable behavioral pattern. Therefore, ANN structures can be reused or associated to provide a similar meaning within the network.

2.5.4.3 Prediction Problems

According to Kasabov (1996), prediction problems intend to make an ANN perform mapping from k time steps before a time step t of the variable \underline{x} to the next n time steps from the variable values of \underline{y} , starting with the future time step $t + 1$. Here, k is often referred to as *lag of the prediction* and n is referred to as *prediction width*.

Principally, \underline{y} can follow the same structure as \underline{x} . If we have various variables $x_1, x_2, x_3, \dots, x_d$ of the *moments* \underline{x}^t from the time step $(t - k)$ to t that are $\underline{x}^{t-k}, \underline{x}^{t-k+1}, \dots, \underline{x}^{t-1}, \underline{x}^t$ and jointly considered as *input sequence* $[\underline{x}]^m$, the prediction task of the ANN refers to the production of *future moments* \underline{y}^t from the time step t to $(t + 1)$, with $\underline{y}^{t+1}, \underline{y}^{t+2}, \dots, \underline{y}^{t+n}$, providing each of the elements $y_1, y_2, y_3, \dots, y_k$ and jointly considered as *output sequence* $[\underline{y}]^m$.

In the learning phase, the parameters of ANN are determined from a finite set

$$S = \{([\underline{x}^{t-k}]^m, [\underline{y}^{t+n}]^m) | \underline{x}^m \in \mathbb{R}^d, \underline{y}^m \in \mathbb{R}^d, m = 1, \dots, M\}. \quad (2.17)$$

This means, that each feature vector \underline{x}^m from the time steps $t - k$ is labeled with its future vector \underline{y}^m from the time steps $t + n$. During the *recall phase*, when the learning has finalized successfully on m examples, the ANN can be faced with unlabeled observations \underline{x} such that it estimates their future \underline{y} .

Depending on the codification of the input vector elements $x \in X$ and the target vector elements $y \in Y$, these can represent all real values, classes, or a mixture of the two and the following forms of prediction problems can be identified.

Sequence Prediction: The first problem formulation refers to the learning task of producing the next sequential value, when an input sequence is perceived (Sun and Giles, 2001; Begleiter, El-Yaniv, and Yona, 2004; Aggarwal, 2014; Gueniche et al., 2015). Here, $x_i \in \mathbb{R}^d$ and $y \in \mathbb{R}$.

Sequence Classification: The second problem formulation refers to the circumstance in which the classification problem (section 2.5.4.2) considers a sequence as input for which a class label needs to be predicted (Eck and Schmidhuber, 2002a; Eck and Schmidhuber, 2002b; as well as Aggarwal, 2014). Here, prediction problems intend to make an ANN perform mapping from a continuous input space \mathbb{R}^d into a

finite set of classes $Y = \{1, \dots, L\}$ of the next time step $t + 1$, over an increasing or decreasing stock course. If the class memberships $\{y \in Y\}$ are encoded through a 1-of-L coding into a binary vector $\underline{z} \in \{0, 1\}^L$, the number of trend classes will lead to the number of output neurons. Building on this encoding scheme, the learning tasks realizes $\mathbb{R}^d \mapsto \mathbb{R}^L$.

Others differentiate further in *sequence recognition* and *sequential decision making*. While the first focuses on the prediction of a sequence belonging to a class of legitimate sequences according to some criteria, the second predicts (a sequence of) actions to accomplish a goal, follow a trajectory, or maximize a reward or minimize costs (Sun and Giles, 2001).

Sequence Generation: The third problem formulation refers to the task to produce an output sequence given an input sequence (Graves, 2013a; Vinyals, 2014). Since the new output sequence has the same general characteristics as other sequences in the corpus, one speaks from the generation of an individual sequence. Here, $x_i \in \mathbb{R}^d$ and $y \in \mathbb{R}^d$.

Sequence to Sequence Prediction: The fourth problem formulation refers to the task to produce an output sequence when the ANN is faced with an input sequence that continues the input sequence (Sutskever, Vinyals, and Le, 2014; as well as Luong et al., 2015). Here, $x_i \in \mathbb{R}^d$ and $y \in \mathbb{R}^d$.

More Complex Predictions: Since the variables subject to prediction \underline{y} (dependent variables) can be differentiated from the past data variables \underline{x} (independent variables) and both are operating on the time scale, one can consider prediction problems as function approximation tasks focusing on the identification of a regression function for temporal relations (Backhaus et al., 2008; Nauck et al., 2013, p. 36). Here, one assumes to see patterns reflected in historic data that occur again in the future. Hence, the learning problem can provide a mixture of categorical and real values.

Here, as the number of elements d of \underline{x} equals the number of elements of \underline{y} , which is referred to as k , we have $l = k = 1$ which form *univariate* or *simple* prediction problems. When $l > 1$ and $k = 1$, prediction problems are considered as *multivariate* or *multiple* prediction problems. If there are multiple dependent variables $k > 1$, these are denominated as *multi-target* prediction (MTP) problems (Waegeman, Dembczyński, and Hüllermeier, 2019).

Interim Conclusion: As one considers supervised temporal predictions, and the sensory inputs and the sensory outputs follow the same structure, a neuronal output of time step t can serve as neuronal input for time step $t + 1$. So, a repetitive activation by

the ANN itself can function as an independent simulator. Hence, this is particularly desirable for the construction of a CoNM since this enables the following: first, it allows the integration of real sensory input and predicted values; second, it supports the building of expectations as one sees the predicted values; third, it enables the controlling of systems as deviations from real and predicted occurrences; and fourth, it enables the derivation of measurements that can be realized in reality when needed.

2.5.5 Neuronal Networks

Originally, artificial neuronal networks (ANNs) were developed in order to model the biological functionality of brains. First attempts can be found in work by Rosenblatt (1963), Rumelhart, Hinton, and Williams (1986), and McCulloch and Pitts (1988), which state that the brain composes of numerous nerve cells, so-called *neurons*, that are connected by *synapses* and *dendrites*. While some authors distinguish analog signal transportation over short distances by dendrites and digital signal transportation over long distances, and some consider a chemical and electrical signal transmission, others do not capture differences and pragmatically consider only one connection among neurons (Lämmel and Cleve, 2012, p. 192). Their synapse and dendrite diameters and electric and chemical transmission rates can be considered as factors that lend *weight* to the connection. As signals are sent from neuron to neuron within the biological brain, a mathematical activation of a neuron represents its average fire rate with similar potentials along this connection.

Since the human brain can establish very complex constellations of neurons, various kinds of ANNs have evolved. These can be categorized with the aid of available behavioral arrangement of synapses. ANN that do not have cyclic connections are referred to as *feedforward networks*. Here, electric currents are not cycling through the same neuron. Examples of the same can be found among the Perceptron (Rosenblatt, 1958), Multilayer Perceptrons (Rumelhart, Hinton, and Williams, 1986; Werbos, 1988; and Bishop, 1995), and Radial Basis Function networks (Broomhead and Lowe, 1988). In contrast, networks that have cyclic connections are referred to as *feedbackward networks* or *recurrent networks*. For examples, please refer to Elman and Jordan networks (Lämmel and Cleve, 2012) and LSTM blocks (Hochreiter and Schmidhuber, 1997a). Further, ANN can be categorized on behalf of the spatial arrangement of neurons available. Examples for the same can be found in Convolutional networks (Schwaiger and Steinwendner, 2019, p. 187), Hopfield networks (Hopfield, 1982), and self-organizing maps and Kohonen maps (Kohonen, 1989), as well as neuronal gas (Lämmel and Cleve, 2012).

Considering ANNs as models (Def. 1) that have a processual meaning (Def. 3), the following considers ANNs as processual models:

Definition 24 (Neuronal Model).

A neuronal model (synonym for artificial neuronal network) is a mathematical representation x (denominated as model system) of some part of an object system y , which is the human brain. The part being delimited by a system boundary sb consists of, first, a set of system elements se or the so-called neurons, and second, neuronal networks as individual sub-systems ss which all are connected by a set of relations sr , the so-called weights representing dendrites and synapses, all defined by the interpretation of a model creator mc for the model user mu for description, simulation, analysis, design, and evaluation purposes p on behalf of a reproduction of the functioning of the human brain, and a processual meaning within the time period t . It is constructed through construction processes following a consistent formalization f .

The following focuses only on the explanation of ANN mechanisms that are relevant for the construction of the CoNM. This starts with the functioning of an individual neuron as a basic unit for ANN. Further explanations consider a joint operation of various neurons, which functions as a basic network structure known as the so-called Multilayer Perceptron. To complement the Multilayer Perceptron, recurrent networks will follow.

2.5.5.1 Single Neurons

A single neuron, or single layer perceptron, can be interpreted as a mapping of an input to an output. For this, an *activation function* is applied. As Fig. 2.50 shall show, the input signal a^t of time step t is transferred to the output signal b^t by the application of the activation function $\theta(\dots)$. At the bottom, one can see a collection of neuron visualizations with the most common activation functions. Since neuronal models presented here only provide feedforward connections, they do not establish a dynamic behavior. This is why the following equations do not show time-dependent variables, although single neurons are able to deal with them. Time-dependent formulations will be introduced following section 2.5.5.3.

These activation functions can be chosen with flexibility so that they fit the task of the single neuron. Often, they are non-linear and differentiable so that further calculations are simplified. During learning procedures, an error arising at the output of the neuron, called the ϵ , is transformed with the aid of the neuron-specific activation function to the error called δ that can be found at the entrance of the single

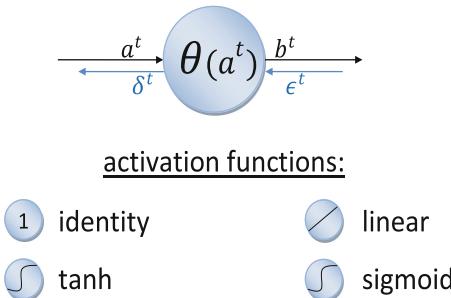


Figure 2.50 A single neuron

neuron. This is why the following does not only specify activation functions, but also provides corresponding derivations.

The two most common activation functions θ are called the hyperbolic tangent

$$\theta(a) = \tanh(a) = \frac{e^{2a} - 1}{e^{2a} + 1}, \quad [-1, +1] \quad (2.18)$$

$$\frac{\partial \theta(a)}{\partial a} = 1 - \tanh(a)^2 \quad (2.19)$$

and the logistic sigmoid function

$$\theta(a) = \sigma(a) = \frac{1}{1 + e^{-a}}, \quad [0, +1] \quad (2.20)$$

$$\frac{\partial \theta(a)}{\partial a} = \sigma(a) \cdot (1 - \sigma(a)). \quad (2.21)$$

The choice between Eq. 2.18 and Eq. 2.20 as an activation function is not essential for learning since they are equivalent by the linear transformation $\tanh(a) = 2\sigma(2a) - 1$. The logistic sigmoid function can be further centered as follows:

$$\theta(a) = 2 \cdot \sigma(a) - 1 = 2 \cdot \frac{1}{1 + e^{-a}} - 1, \quad [-1, +1] \quad (2.22)$$

$$\frac{\partial \theta(a)}{\partial a} = 2 \cdot [\sigma(a) \cdot (1 - \sigma(a))]. \quad (2.23)$$

Mostly, bias neurons provide a constant function as activation functions such that they supply a constant value, probably 1, continuously. Based on the knowledge codified within the ANN, and dependent on the weight of the bias neuron and consecutive neurons, a bias signal can be used without variation in order to represent an offset:

$$\theta(a) = 1 \quad (2.24)$$

$$\frac{\partial\theta(a)}{\partial a} = 0. \quad (2.25)$$

Input neurons provide an identity as a transfer function as they intend to route signals without affecting them:

$$\theta(a) = a \quad (2.26)$$

$$\frac{\partial\theta(a)}{\partial a} = 1. \quad (2.27)$$

Further alternatives for activation functions can be found in the rectified linear unit function (ReLU), that was built so that the particular phenomenon of deep networks having only small weight adjustments in their first layers, called *saturation*, can be overcome:

$$\theta(a) = x^+ = \max(0, a) \quad (2.28)$$

$$\frac{\partial\theta(a)}{\partial a} = \begin{cases} 0, & \text{if } a < 0 \\ 1, & \text{if } a \geq 0. \end{cases} \quad (2.29)$$

As the ReLU is unsteady at 0, and neurons can enter a state of not further participating in the learning process, the leaky ReLU function was built (Schwaiger and Steinwendner, 2019, p. 201):

$$\theta(a) = \max(\alpha a, a), \quad \alpha \text{ is very small} \quad (2.30)$$

$$\frac{\partial\theta(a)}{\partial a} = \begin{cases} \alpha, & \text{if } a < 0 \\ 1, & \text{if } a \geq 0. \end{cases} \quad (2.31)$$

Such an exponential linear unit function (ELU) converges even faster than the ReLU (Schwaiger and Steinwendner, 2019, p. 202):

$$\theta(a) = \begin{cases} \alpha(e^a - 1), & \text{if } a < 0 \\ a, & \text{if } a \geq 0 \end{cases} \quad (2.32)$$

$$\frac{\partial \theta(a)}{\partial a} = \begin{cases} \alpha e^a, & \text{if } a < 0 \\ 1, & \text{if } a > 0. \end{cases} \quad (2.33)$$

2.5.5.2 Multilayer Perceptrons

Given below is an ANN with I input neurons in the input layer, several neurons in various hidden layers, and K output neurons in the output layer. This refers to a combination of network elements as they have been characterized in section 2.5.5.1. An example of an ANN of this sort can be seen in Fig. 2.51. Since neurons are only connected unidirectionally, they are categorized to *feedforward networks*. As they provide various layers, they are referred to as Multilayer Perceptrons (MLPs). Since single neurons are able to approximate simple, one-dimensional relations, various neurons of one layer can jointly deal with several dimensions. On behalf of several layers, complex and multi-dimensional relations can be approximated. Based on universal approximation theory of Hornik, Stinchcombe, and White (1989), ANNs have proven to be able to approximate any function at an arbitrary accuracy. For this, either a sufficient number of neurons in various hidden layers or a sufficient number of neurons in one single neuronal layer is required.

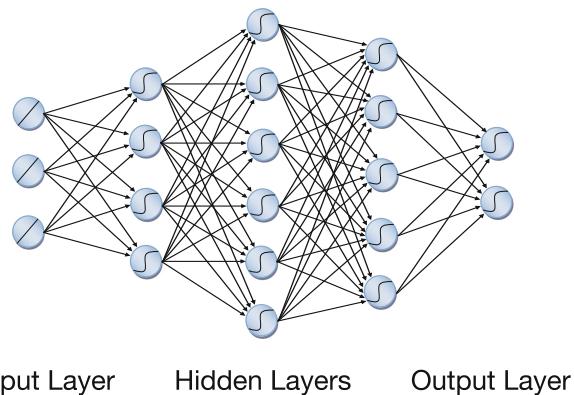


Figure 2.51 Example of a Multilayer Perceptron

As these kinds of networks only provide feedforward connections, they realize approximations without changing their behavior over the course of time. This is why the following equations do not show time-dependent variables yet, although

MLPs are able to deal with them. Time-dependent formulations will be introduced from section 2.5.5.3.

Input neurons are stimulated by the input vector \underline{x} , which contains the elements x_i and serves as container for input signals. Since input neurons do not affect the input and just route this signal (cf. Eq. 2.26), signals reaching a neuron of the first hidden layer a_h can be identified with the aid of Eq. 2.34. Here, w_{ih} represents the connection of the input neuron i and neuron h of that hidden layer. This signal will be transferred by the activation function θ_h of that neuron of the hidden layer so that its output signal b_h can be derived with Eq. 2.35.

$$a_h = \sum_{i=1}^I w_{ih} x_i \quad (2.34)$$

$$b_h = \theta_h(a_h) \quad (2.35)$$

This calculation will be repeated for all neurons of that hidden layer. Then, output signals of the current hidden layer serve as input signals of the consecutive hidden layer. So, calculations of Eq. 2.34 and Eq. 2.35 can be applied for this and any subsequent layers without any change until the output layer is reached. The last output signal b_h can be interpreted as result of the joint ANN. Hence, it is considered as b_k . Since calculations are realized from the first to the last layer, in order to identify the resulting signal, these steps are referred to as a *forward pass*.

Since MLPs have been realized as differentiable constructs, they can be trained by attempts to minimize a differentiable error function. Example functions can be found in section 2.5.6.4. In order to carry out corresponding optimizations, *gradient descent algorithms* are applied that are described in section 2.5.6.1. In the simplest form, in the first step, all error gradients must be identified which are derived by the connection weights. In the second step, a weight adjustment can be derived by the negative increase of the gradients. An efficient method to identify any gradients within a network refers to the famous *backpropagation* (Rumelhart, Hinton, and Williams, 1986; Werbos, 1988; and Williams and Zipser, 1995). Since gradients are identified by the propagation of errors from the back to the front, the following presents relevant steps as a *backward pass*. This will be the algorithmic foundation of the CoNM.

The proceeding is illustrated by the following: an error signal is injected at the very last layer of a neuronal network and transferred from the back to the very beginning of the neuronal network. For this, the same weight connections and activation functions are applied that have been used for the forward pass. The error

that is injected at the output neurons is referred to as E^T and is concretized in section 2.5.6.1. The outgoing error ϵ_k of neuron $k \in K$ is, therefore, determined by

$$\epsilon_k = \frac{\partial E^T}{\partial b_k}. \quad (2.36)$$

Since the output signal was affected by the activation function of that neuron, the respective error signal is affected so that the ingoing error δ_k refers to

$$\delta_k = \epsilon_k \cdot \frac{\partial b_k}{\partial a_k}. \quad (2.37)$$

The influence of the preceded weights is considered as follows:

$$\frac{\partial a_k}{\partial w_h}. \quad (2.38)$$

By the joint interplay of Eq. 2.36–2.38, the failure gradient ∇_{hk} between the output neurons and neurons of the hidden layer can be derived. Using this, the weight adjustment and the optimization is carried out.

$$\nabla_{hk} = -\frac{\partial E^T}{\partial b_k} \cdot \frac{\partial b_k}{\partial a_k} \cdot \frac{\partial a_k}{\partial w_h} = -\delta_k b_h \quad (2.39)$$

This will be repeated for any neurons of the output layer. Then, the ingoing errors of the current hidden layer can stand as outgoing errors of the preceding layer and the current hidden layer is interpreted as the output layer for neurons of preceding layers. So, Eq. 2.39 can be applied again. This proceeding is repeated until the current hidden layers equals the input layer and all gradients of the entire network have been determined. With them, the weight adjustment can be carried out and key insights are reflected within the neuronal weights.

2.5.5.3 Recurrent Neuronal Networks

In extension to the previously kinds of networks, *recurrent neuronal networks* (RNNs) provide closed connection cycles (Fig. 2.52). Here, one can find a MLP which provides a hidden layer having two neurons. These two neurons provide recurrent connections with themselves, visualized in a bright red, and with foreign neurons of that layer, visualized in dark red. The corresponding weight is referred to as $w_{h'h}$.

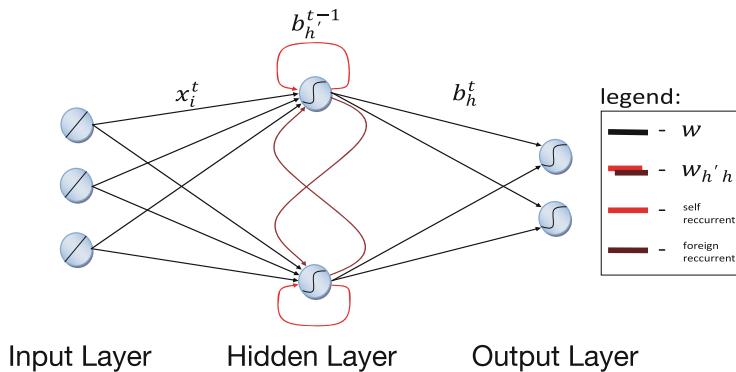


Figure 2.52 Example of a Recurrent Neuronal Network

Since recurrent connection characterize connections among neurons of the same layer and consecutive layers, neurons are fed with their current or consecutive outputs and the ANN changes its behavior over the course of time, depending on what they have seen before. Past activations are considered in addition to current activations so that the current network output is affected by the network's history. One can say that the network is *remembering* its perception history and thus, is particularly suited for the learning of temporal relations. Hence, the following equations consider time-dependent variables and focus on the learning of time series.

Parallel to the common approximation theory of MLPs, recurrent ANNs are able to approximate temporal relations by arbitrary accuracy if the hidden layer provides a sufficiently high number of neurons having recurrent connections (Hammer, 2000).

The calculations within recurrent ANNs are quite similar to MLP calculations. The difference lies in the calculation of the incoming activation of neurons of the hidden layer h at time step t . This considers the following: first, the activation of input neurons of time step t ; and second, the outgoing activation of neurons of the hidden layer of the previous time step $t - 1$.

$$a_h^t = \sum_{i=1}^I w_{ih} x_i^t + \sum_{h'=1}^H w_{h'h} b_{h'}^{t-1} \quad (2.40)$$

Further, difference lies in the backward pass such that additional calculations steps have to be realized in the backward propagation:

$$\delta_h^t = \theta'(a_h^t) \left(\sum_{k=1}^K \delta_k^t w_{hk} + \sum_{h'=1}^H \delta_{h'}^{t+1} w_{h'h} \right). \quad (2.41)$$

The commonly known algorithm called *Backpropagation Through Time* realizes the systematic determination of gradients required for learning in recurrent ANNs (Werbos, 1990; Williams and Zipser, 1995).

2.5.6 Training of Neuronal Networks

The following section shows how the neuronal learning process is realized. Since ANNs commonly learn until a certain success criteria is satisfied, the learning process is referred to as *training*.

In order to characterize the training of an ANN, section 2.5.6.1 describes the learning method on behalf of gradients that are derived from errors. For this, various target functions are available and presented in section 2.5.6.4. Requirements for an successful learning are presented in section 2.5.6.5 and section 2.5.6.6. *Successful learning* then is specified in section 2.5.6.7.

2.5.6.1 Error-based Learning Algorithms

As ANNs have been designed as differentiable constructs, they can be trained in order to minimize the error of ANN results $\underline{\mathbf{E}}^T$ of an differentiable error function $e(\mathbf{b}, \mathbf{t})$. Variants of this functions will be defined in section 2.5.6.4. In order to realize corresponding optimizations, *gradient descent algorithms* are used in the form of *batch or online learning*.

2.5.6.1.1 Algorithmic Variants

In the simplest form, weight adjustments are realized on behalf of gradients that are determined in dependency of the network's weights $\underline{\mathbf{w}}$ of learning iteration n .

$$\Delta \underline{\mathbf{w}}(n) = - \frac{\partial \underline{\mathbf{E}}^T}{\partial \underline{\mathbf{w}}(n)} \quad (2.42)$$

As the error is propagated from the back to the front of an ANN, the algorithmic approach is known as *backpropagation* and requires the following steps:

1. Input patterns are fed into the ANN and propagated through all layers.
2. The network activation is compared with the intended activation so that an error is derived.
3. The error of the output layer is propagated from the very last layer to the very first layer so that, for each layer, weight adjustments can be derived.
4. Weight adjustments are carried out.

A suitable metaphor for the process of weight adjustments can be trekking in the mountains. Here, the mountains represent errors being produced by the network. The objective of the trekker is to locate the deepest valley (global optima). In accordance with Eq. 2.42, the trekker moves during the course of each learning iteration having a fixed step length in the opposite direction of the error gradient. So, it approaches an error valley.

In order to increase their chances of locating the global optima, and reducing the chances of being stuck in a local error valley, various creative approaches can be identified. The following presents the two most prominent approaches in detail since they will serve as the foundation for the CoNM. Others are only presented in an overview.

Learning Rate Variants: As a first variation, the weight adjustment presented by Eq. 2.42 is relativized by a *learning rate* α . Picking up the metaphor introduced, the trekker is allowed to vary their step length by modifying the learning rate.

$$\Delta \underline{\mathbf{w}}(n) = -\alpha \frac{\partial \underline{\mathbf{E}}^T}{\partial \underline{\mathbf{w}}(n)}, \quad \text{where } 0 \leq \alpha \leq 1 \quad (2.43)$$

The bigger the α , the bigger the weight adjustment in regard with the current error gradient. Hence, the choice of α is essential for learning success and the question of whether a global optima can be located. If it is too big, the errors produced will tend to oscillate around an optima without reaching it. If it is set too small, the trekker will tend not to overcome the local optima and the global optima is not identified at all. In a perfect world, α adapts to the current learning situation. For instance, it is initialized by a big value so that it approaches the global optima quickly and is able to overcome local optima. Afterwards, it is reduced so that it will not tend to oscillate and the global optima is reached.

Momentum Variants: As a second variation, a kind of momentum is introduced. Following the metaphor of the trekker, the movement is modeled to be inertial. Besides the current weight adjustment $\Delta \underline{\mathbf{w}}(n)$, historic weight adjustments

$\Delta \underline{\mathbf{w}}(n - 1)$ are considered. This shall reduce the influence of local optima since the effect of gradients will increase if they do not change rapidly. The proportion of historic weight adjustments is determined by the factor m as follows:

$$\Delta \underline{\mathbf{w}}(n) = m \Delta \underline{\mathbf{w}}(n - 1) - \alpha \frac{\partial \underline{\mathbf{E}}^T}{\partial \underline{\mathbf{w}}(n)}, \quad \text{where } 0 \leq m \leq 1. \quad (2.44)$$

The bigger the m set, the bigger the influence of historic weight adjustments. With this, the current weight adjustments will be more inertial. Hence, the choice of m is also essential for learning success.

Variants Overview: Experiments about the design of gradient procedures were first realized in the 1990s (Plaut, Nowlan, and Hinton, 1986; Bishop, 1995). By now, various improved versions are available, such as PROP (Riedmiller and Braun, 1993), Quickprop (Fahlman, 1989), conjugated gradients (Hestenes and Stiefel, 1952; Shewchuk, 1994), and L-BFGS (Byrd et al., 1995). In an LSTM-based ANN, Eck and Schmidhuber (2002a) consider a combination of the RTRL approach of Robinson and Fallside (1987) and the BPTT approach of Williams and Zipser (1995). Here, they deal with approximated error gradients and shorten the time between each time step (Eck and Schmidhuber, 2002a; Eck and Schmidhuber, 2002b). Although the CoNM will use ANNs similar to Eck and Schmidhuber, error gradients will be derived according to Eq. 2.44. As such, the CoNM follows the approach of Graves and Schmidhuber (2005) that have successfully considered this in the context of LSTM-based ANNs.

2.5.6.1.2 Batch and Online Learning

If gradient descent algorithm variants are applied after having seen all samples of a training set S , this results in *batch learning*. Alternatively, these algorithmic variants are applied after having seen one sample of that training set, which is called *online learning*. According to LeCun et al. (1998), the latter is more efficient and more robust against local optima.

As the CoNM intends to carry out an ongoing continuous learning inspired by the CIP from the knowledge management domain (section 2.4.2.2), CoNM-based training attempts refine online learning. As the CoNM intends to evaluate the ANN within its global context, according to Grum et al. (2018), over scenarios and system alternatives, the CoNM-based testing attempts refine batch learning. Hence, the following specifies both, errors in online learning contexts and errors in batch learning contexts.

Errors in Online Learning: Assume K to represent the total number of neurons within the output layer, T to represent the total number of data input signals forming a sequence, and the error injected at output neuron represented by k and time step t , which is called E_k^t and follows a certain target function (cf. section 2.5.6.4). Thus, the vector version of the corresponding training error $\underline{\mathbf{E}}^T$ can be derived. This will be used in gradient descent algorithms as online learning as follows:

$$\underline{\mathbf{E}}^T = \eta \cdot \sum_{t=1}^T \underline{\mathbf{E}}^t. \quad (2.45)$$

The scalar version of this errors refers to the following:

$$E^T = \eta \cdot \sum_{k=1}^K \sum_{t=1}^T E_k^t. \quad (2.46)$$

Here, η represents any kind of transformations, such as normalizations (the division by K and T , etc.; cf. section 2.5.6.6) or as compensation for weighted samples.

Errors in Batch Learning: The training error for batch learning E^T can be derived by the aggregation of errors of the sequence. Here, η represents the division by K , T , the number of sequences (normalization), and the compensation of weighted samples:

$$\underline{\mathbf{E}}^T = \eta \cdot \sum_{seq=1}^{Seq} \sum_{t=1}^T \underline{\mathbf{E}}_{seq}^t. \quad (2.47)$$

The scalar version of this errors refers to the following:

$$E^T = \eta \cdot \sum_{seq=1}^{Seq} \sum_{k=1}^K \sum_{t=1}^T E_{k_{seq}}^t. \quad (2.48)$$

Besides use in gradient descent algorithms and for learning purposes, these kinds of error KPIs can be used in order to evaluate the performance of the ANN while considering training data, the performance for generalization while considering test data and validation data, and the relativization of the meaning of the inner working of the ANN.

2.5.6.2 Competitive Learning Algorithms

As it is intended to consider spatial relations during the course of learning, which is not considered by error-based learning algorithms, a competitive learning approach is ideal. The principle applied in competitive neuronal networks recodes sets of correlated inputs in a way that they fit to one of a few output neurons. So, the principle removes the redundancy in representation and supports the interpretability of a certain neuronal region.

It is biologically plausible since it models the phenomenon of *lateral inhibition* and *local reception fields*. Examples can be found in sensory stimuli which are positioned on the skin or the visionary perception at the retina, among others (Barlow, 1989; Ritter, Martinetz, and Schulten, 1991, p. 68; as well as Rolls and Deco, 2001), from which an unsupervised learning is realized.

There are two kinds of competitive learning approaches (Lämmel and Cleve, 2012, p. 251) as follows: first, *hard competitive learning* that only affects weights which are directly dependent on the winning neuron; and second, *soft competitive learning* that affects more weights than the winning neuron and its close neighbors. According to the softness defined, it can affect any neuron available in the ANN, although a focus still lies on the winning neuron and its neighbors.

The weight adjustment of hard competitive learning is carried out as follows (Lämmel and Cleve, 2012, p. 256, 258):

$$\Delta w_{i,j} = \begin{cases} \alpha \cdot h_{j,z} \cdot (x_i - w_{i,j}), & \text{if } dist(j, z) \geq r \\ 0 & \text{otherwise} \end{cases} \quad (2.49)$$

having

$$h_{j,z} = e^{-\frac{dist(j,z)^2}{2r^2}}. \quad (2.50)$$

Here, $w_{i,j}$ refers to the weight of neurons i of a first layer, e.g., the input layer, to neurons j of the consecutive layer, e.g., the hidden layer. While α refers to the learning rate and controls the strength of the modification (similar to Eq. 2.43), and x_i refers to the input vector of the first layer, $dist(j, z)$ refers to the distance of neuron j to the winning neuron z of the consecutive layer which is modeled with the aid of the Gaussian bell-shaped curve and results in $h_{j,z}$. The parameter r controls the range of the bell-shaped curve so that it is an ideal wide bell-shaped curve used for the beginning of the learning (big r -values) since many neurons are affected and the most promising neuronal location within the network is detected. Further, it is ideal to use smaller big r -values for later learning iterations because neuronal

regions already have specialized during the course of learning and a specialization focuses on expert neurons.

A soft version of the competitive learning considers weight adjustments $w_{i,j}$ similar to Eq. 2.49, but it brings neurons in a running order of f_1, f_2, \dots, f_N , for which $dist_{f_1}(j, z) \leq dist_{f_2}(j, z) \leq \dots \leq dist_{f_N}(j, z)$ holds and weight adjustments for all neurons available within the ANN dependent on that ordering are considered. Hence, the learning is carried out as follows (Lämmel and Cleve, 2012, p. 265):

$$\Delta w_{i,j}^{f_k} = \alpha_t \cdot h(t, k) \cdot (x_i - w_{i,j}^{f_k}), \text{ for } k = 1 \dots N \quad (2.51)$$

having

$$\alpha_t = \alpha \cdot \left(\frac{\alpha_{t_{max}}}{\alpha_0} \right)^{t/t_{max}}, \text{ with } \alpha_0 > \alpha_{t_{max}}, \quad (2.52)$$

$$\varepsilon_t = \varepsilon \cdot \left(\frac{\varepsilon_{t_{max}}}{\varepsilon_0} \right)^{t/t_{max}}, \text{ with } \varepsilon_0 > \varepsilon_{t_{max}}, \text{ and} \quad (2.53)$$

$$h(t, k) = e^{(\frac{1-k}{\varepsilon_t})}. \quad (2.54)$$

Here, α refers to the learning rate (similar to Eq. 2.43) and x_i refers to the input vector of one layer. The influence of the distance between neuron j and the winning neuron z of the consecutive layer is modeled with the aid of the Gaussian bell-shaped curve resulting in $h(t, k)$, which considers all neurons depending on the running order identified. Here, the influence is reduced by a greater distance indicated by higher k . Further, the influence is reduced by later learning iterations t such that the learning rate α_t and the Gaussian bell-shaped curve width ε_t are dependent on t .

2.5.6.3 Hebbian Learning Algorithms

Introduced by Donald Olding Hebb in 1949, an attempt to explain synaptic plasticity is provided (Hebb, 1949) that claims that an increase in synaptic efficacy arises from a presynaptic cell's repeated and persistent stimulation of a postsynaptic cell. Known under various names, such as *Hebbian theory*, *cell assembly theory*, *Hebb's postulate*, and *Hebb's rule*, the adaptation of brain neurons during the learning process is modeled by the following:

$$\Delta w_{i,j} = \eta \cdot a_i \cdot a_j. \quad (2.55)$$

Here, $w_{i,j}$ represents the weight of the connection from neuron i to neuron j . Further, a_i refers to the activation of neuron i and a_j refers to the activation of

neuron j , which is connected by neuron i . Hence, the more neuron i is activated jointly with neuron j , the more they will react sensibly on each other.

Since it can be shown that, for any neuron model, Hebb's rule is unstable (Principe, Euliano, and Lefebvre, 1999), variations refer to Oja's rule (Oja, 1982), BCM theory (Bienenstock, Cooper, and Munro, 1982), and the generalized Hebbian Algorithm known as *Sanger's rule* (Sanger, 1989).

2.5.6.4 Target Functions

Every target function establishes their own error characteristic and causes a proper learning behavior of the ANN. Since the target function is considered in gradient descent algorithms (cf. section 2.5.6.1), the choice of a specific target function directly influences the individual learning performance. The cause for this lies in the establishment of an individual error mountain by the target function $e(\dots)$ and its gradient $e(\dots)'$. As a result of the target function, the error $\underline{\mathbf{E}}^t$ is determined, which represents the error elements E_k^t and represents the error produced at neurons $k = 1, 2, \dots, K$ of the output layer at time step t .

$$\underline{\mathbf{E}}^t = e(\underline{\mathbf{b}}^t, \underline{\mathbf{t}}^t) \quad (2.56)$$

$$(\underline{\mathbf{E}}^t)' = \frac{\partial e(\underline{\mathbf{b}}^t, \underline{\mathbf{t}}^t)}{\partial \underline{\mathbf{b}}^t} \quad (2.57)$$

Here, $\underline{\mathbf{b}}^t$ represents the prediction result of the ANN at time step t , which holds the individual result elements b_k^t for the output neurons $k \in K$. Dependent on the current target function, the result vector $\underline{\mathbf{b}}^t$ is compared with the target vector $\underline{\mathbf{t}}^t$, which represents the intended prediction of time step t and contains the target elements t_k^t for the output neurons $k = 1, \dots, K$.

In order to create a spatial impression of the target function's individual characteristics, each target function definition is accompanied with a visualization of its error mountain at time step t and corresponding gradients in two dimensions. The height of the error mountain can be interpreted as the scalar version of the error E^t defined in Eq. 2.46 and Eq. 2.48.

In the following, target functions that are very well known, very often used, and perform quite well on a great range of different kinds of learning problems are shown. This is why they are referred to as *base types*. For the construction of a CoNM, they serve as the foundation for the technical realization and result in learning agents that are interpreted as *machine learners* solving individual learning tasks.

2.5.6.4.1 Mean Square Error

The mean square error E_{MS} is used very often because of its very simple calculation, its very simple derivation, and its characteristics as it is more robust than outliers, which are observations that lie outside the overall pattern of a distribution. This results in a very easy handling of, for instance, backpropagation algorithms. It is analyzed very well in all kinds of literature and considered state of the art (Kline and Berardi, 2005; as well as Golik, Doetsch, and Ney, 2013).

$$e(\underline{\mathbf{b}}^t, \underline{\mathbf{t}}^t) = 0.5 \cdot (\underline{\mathbf{t}}^t - \underline{\mathbf{b}}^t)^2 \quad (2.58)$$

$$\frac{\partial e(\underline{\mathbf{b}}^t, \underline{\mathbf{t}}^t)}{\partial \underline{\mathbf{b}}^t} = \underline{\mathbf{b}}^t - \underline{\mathbf{t}}^t \quad (2.59)$$

Fig. 2.53 visualizes the vector version of the mean square error \underline{E}_{MS}^t and its derivative $(\underline{E}_{MS}^t)'$ at a certain time step t with the help of two example dimensions b_1^t and b_2^t , while the target is $\underline{\mathbf{t}}^t = (1.0, 0.0)^T$. Clearly, one can see the corresponding quarter of a paraboloid of revolution (a), where the ascent of the error increases linearly with the increase of its errors. Also, one can see its derivative (b), which shows the biggest gradient at that point, where the decimal results are completely wrong, and the smallest gradient at that point, where the decimal results are completely correct. In between, one can see that the gradients increase visibly with increasingly wrong decimal results in contrast to the derivative of \underline{E}_{CE}^t defined in Eq. 2.61 (see Fig. 2.54).

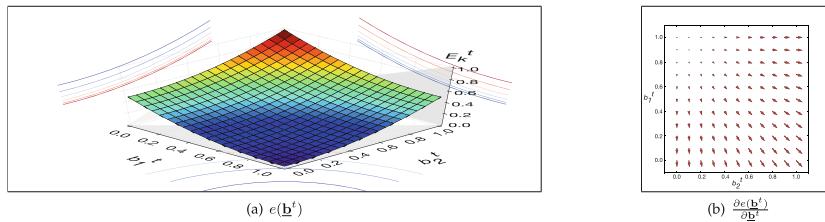


Figure 2.53 Mean Square Error E_{MS}^t for $\underline{\mathbf{t}}^t = (1.0, 0.0)^T$

2.5.6.4.2 Cross Entropy Error

Cross entropy error E_{CE} is used very often because it leads to faster convergence and better results with respect to classification error rates, which results in a very easy handling of, for instance, backpropagation algorithms. It is analyzed very well in all kinds of literature and considered state of the art (Kline and Berardi, 2005; as well as Golik, Doetsch, and Ney, 2013).

$$e(\underline{\mathbf{b}}^t, \underline{\mathbf{t}}^t) = -\underline{\mathbf{t}}^t \cdot \ln(\underline{\mathbf{b}}^t) - (1 - \underline{\mathbf{t}}^t) \ln(1 - \underline{\mathbf{b}}^t) \quad (2.60)$$

$$\frac{\partial e(\underline{\mathbf{b}}^t, \underline{\mathbf{t}}^t)}{\partial \underline{\mathbf{b}}^t} = \frac{\underline{\mathbf{t}}^t - \underline{\mathbf{b}}^t}{\underline{\mathbf{b}}^t(1 - \underline{\mathbf{b}}^t)} \quad (2.61)$$

Fig. 2.54 visualizes the vector version of the cross entropy error $\underline{\mathbf{E}}_{CE}^t$ and its derivative $(\underline{\mathbf{E}}_{CE}^t)'$ at a certain time step t with the help of two example dimensions b_1^t and b_2^t , while the target is $\underline{\mathbf{t}}^t = (1.0, 0.0)^T$. Clearly, one can see the corresponding quarter of a *swimming pool* (a). The ascent of the error only slightly increases with the increase of its errors (where the water is located in a swimming pool), but explodes at the *swimming pool borders* since the *logarithm* is not defined at those parts. Also, one can see its derivative (b), which shows the biggest gradient at that point, where the decimal results are completely wrong, and shows the smallest gradient at that point, where the decimal results are completely correct. In between, the gradients only slightly increase with increasing wrong decimal results in contrast to the derivative of $\underline{\mathbf{E}}_{MS}^t$ defined in Eq. 2.59 (see Fig. 2.53).

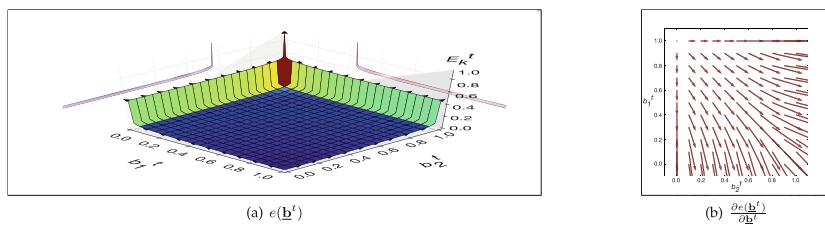


Figure 2.54 Cross Entropy Error E_{CE}^t for $\underline{\mathbf{t}}^t = (1.0, 0.0)^T$

2.5.6.4.3 Threshold Error

The problem itself can be described with the threshold error E_{Th} . Since a classification can either be true or false, the problem needs a threshold which decides if a prediction is part of a class or not. Hence, this error function *categorizes* the result for each neuron and determines the difference to the corresponding target as an error:

$$e(b_k^t, t_k^t) = \begin{cases} |(t_k^t - 1)| & , b_k^t \geq 0.5 \\ |(t_k^t - 0)| & , b_k^t < 0.5 \end{cases} \quad (2.62)$$

$$\frac{\partial e(\mathbf{b}^t, \underline{\mathbf{t}}^t)}{\partial \mathbf{b}^t} = \mathbf{0}. \quad (2.63)$$

As its derivative is always zero, there is no chance to minimize the error for gradient descent algorithms. Fig. 2.55 visualizes the vector version of the threshold error \underline{E}_{Th}^t and its derivative $(\underline{E}_{Th}^t)'$ at a certain time step t with the help of two example dimensions b_1^t and b_2^t , while the target is $\underline{\mathbf{t}}^t = (1.0, 0.0)^T$. Clearly, one can see plateaus at three different heights (a): one at 0.0, when all results are correct; one at 0.5, when only one result is correct; and one at 1.0, when all results are incorrect. Also, one can see its derivative (b) which shows no increase at all.

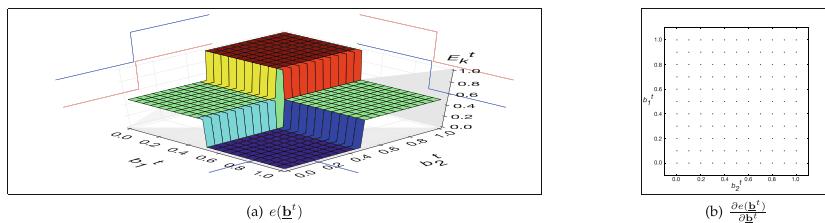


Figure 2.55 Threshold Error E_{Th}^t for $\underline{\mathbf{t}}^t = (1.0, 0.0)^T$

Since this target function is not suited to carry out ANN learning, it can be used in combination with target functions having a gradient so that domain-specific knowledge can be transcoded in the target function.

2.5.6.5 Weight Initialization

Since the number of neurons within an ANN and, with this, the number of connections among them quickly becomes big, the question of which values the connections shall be initialized best arises.

Usually, gradient descent algorithms demand weights to be initialized by random values and to be very small in the range of $[-0.1, 0.1]$. As an alternative, values can follow a Gaussian distribution having the average of 0 and the standard deviation of 0.1.

ANNs part of the CoNM provide connection weights initialized by a normal distribution around 0 and following a standard deviation of 1. This is because of the following reasons: first, this is the standard procedure of the programming library called *PyBrain* (cf. section 2.6.2.2); second, prominent LSTM representatives do not mention the initialization at all (Eck and Schmidhuber, 2002a; Eck and Schmidhuber, 2002b); and third, according to Graves (2012), the choice of the distribution, including the distribution's parametrization for the initialization of a recurrent ANN, does not affect the proper working of the ANN.

2.5.6.6 Input Representation

In order to make an ANN learn an intended prediction result, the underlying dataset must be complete, redundancy-free, and describe a possibly simple context. This is because the ANN tends to learn the simplest possible behavior of a certain context, even if this is not transparent for the dataset constructor. Hence, the design of a dataset and its learning task is essential; section 2.5.2 focuses on the same. Here, according to Graves (2012), ANN experts refer to *black magic* since the data value selection and representation is even more important than the underlying learning algorithm. So, it is recommended by experts to carry out normalization steps as suggested by Joost and Schiffmann (1998): all input data needs to be standardized with regard to the training dataset S (Eck and Schmidhuber, 2002a; Graves, 2012). This means, that any input element x_i of an input vector \underline{x} is transformed so that the average refers to $m_i = 0$ and the standard deviation refers to $\sigma_i = 1$. With the aid of Eq. 2.64, any element x_i can be transformed to a standardized element \hat{x}_i of the vector $\hat{\underline{x}}$.

$$\hat{x}_i = \frac{x_i - m_i}{\sigma_i} \quad (2.64)$$

The average and the standard deviation are determined as follows:

$$m_i = \frac{1}{|S|} \sum_{x \in S} x_i \quad (2.65)$$

$$\sigma_i = \sqrt{\frac{1}{|S|-1} \sum_{x \in S} (m_i - x_i)^2}. \quad (2.66)$$

These normalization steps do not change the data's entropy, but essentially improve their representation for learning agents. So, without mentioning these steps any further, they have been applied in any learning task touched by the CoNM.

2.5.6.7 Generalization

The training foundations presented aid the ANN to generalize relations correctly. This means that the ANN does not learn datasets verbatim (Fig. 2.56 (a)). Here, one can see that the training error E^T is reduced and tends to be 0 on the training data set S . This is caused by almost perfect predictions of the blue data points $(x^t, t^t) \in S$. If the network performance is evaluated with regard to the test dataset S' , a dataset the network has never seen before, its capability to generalize is issued. Here, the generalization error E^G is big because of the bad predictions of the green data points $(x^t, t^t) \in S'$.

This generalization means that an ANN learns to map relations correctly such that possibly small generalization errors are produced by various *validation datasets* and *test datasets*, visualized by Fig. 2.56 (b). Here, one can see that both the E^T and E^G are small because of prudent data representations. So, the ANN is able to produce good predictions of blue training points $\in S$ and green test points $\in S'$.

As one observes the training errors and generalization errors during the course of learning iterations, one can detect when the learning process can be ideally terminated. This is the case at the global optima of the network performance on the test datasets. Further learning iterations support the network's overfitting which is at the expense of generalization. The interplay of training error and generalization error is visualized in Fig. 2.56 (c).

As this figure presents both kinds of errors in an idealized form, the identification of the perfect iteration for stopping the learning procedure is not trivial. Here, testing and speculating is often the only criteria. So, the human interpretation is a major indication as well.

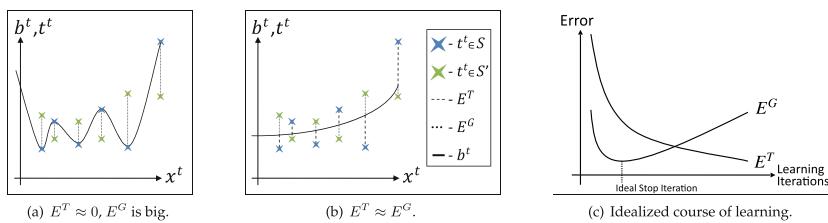


Figure 2.56 Overfitting Compared to Correct Generalization

2.5.7 Network Interpretability and ANN Explainability

Although one can explain the meaning of ANN geometrically, the joint interpretation is very complex and ANNs are perceived as *black boxes* (Kruse et al., 2015, p. 78). The bigger and more complex networks become, the more the limited imaginative power of humans is challenged as this requires a dealing with high-dimensional input spaces. ANNs, therefore, are considered as powerful and mysterious processing units transforming input signals to surprisingly suitable output signals.

Because of a growing number of case studies and application fields showing ANN dominating other techniques, particularly in critical contexts such as health, finance, and security, there is a growing need for humans to understand the inner functioning of neuronal networks. Hence, there are major threads of research focusing on the interpretability of ANN which will be focused in the following.

2.5.7.1 Sensitivity Analyses

Although ANN are valued because of their performance in various kinds of learning problems, they are disadvantaged because, first, they store their tacit knowledge codified by numbers in their connection weights, and second, they store their tacit knowledge codified by numbers in their dynamic behavior. Particularly in larger ANNs, geometric interpretation and explanation is hardly possible because of the limited imagination of humans.

In order to identify the meaning of an individual neuron and, for instance, remove insignificant neurons so that the ANN complexity is reduced, a *sensitivity analysis* can be carried out. With the aid of a sensitivity analysis, the influence of different input signals on the output signal of the ANN is determined. According to Kruse et al. (2015, p. 78), this is realized by the summation of the derivative of the ANN output $out_v^{(l)}$ by the ANN input $ext_u^{(l)}$, which is normalized by the number of patterns $|L_{fixed}|$:

$$s(u) = \frac{1}{|L_{fixed}|} \sum_{l \in L_{fixed}} \sum_{v \in U_{out}} \frac{\partial out_v^{(l)}}{\partial ext_u^{(l)}}, \quad \forall u \in U_{in}. \quad (2.67)$$

Here, $s(u)$ represents the importance of an input for a neuron u for the calculation of the ANN output $out_v^{(l)}$. Based on this, input with small values for $s(u)$ can be removed in order to simplify the ANN structure.

Interim Conclusion: Although the sensitivity analysis issues the importance of individual neuronal connections, the removal of neurons and weights and the refinement of the ANN with further neurons and connections on the basis of knowl-

edge being constructed by the ANN has not been realized till date. Further, current versions of sensitivity analyses focus on the network's current output, which disregards its meaning as knowledge, rather, reinforcing the ANN's current beliefs. Hence, considering knowledge for sensitivity analyses during and after the training goes beyond the context of only considering input features after the training. Extensions, therefore, are ideal for new CoNM algorithms.

2.5.7.2 Feature Visualization

The machine learning community has developed powerful methods that allow the ANN constructor to see how the ANN trained on a data set builds up its understanding over many layers. In general, this is realized on behalf of the generation of example visualizations of the entire network, or parts of a network, that visualize the understanding about a certain input. This is why it is referred to as *feature visualization* (Erhan et al., 2009; Simonyan, Vedaldi, and Zisserman, 2014; Nguyen, Yosinski, and Clune, 2015; Nguyen et al., 2016a; as well as Olah, Mordvintsev, and Schubert, 2017). Considering several feature visualizations, the ANN constructor is able to comprehend how the signal is processed over the network. On behalf of figures from (Olah, Mordvintsev, and Schubert, 2017 under CC-BY 4.0), Fig. 2.57 intends to clarify the process of feature visualization with the aid of the ANN called *GoogLeNet* (Szegedy et al., 2014).

Here, one can see example feature visualizations of nine selected layers of GoogLeNet. While *Layer 3a* deals with simple patterns, that become more sophisticated in *Layer 3b*, after a pooling step, layers start dealing with parts of objects. For example, *Layer 4a* recognizes bookshelves. The *Layer 4b* additionally recognizes contexts. In this case, the neuron responds to trees in front of the sky and ground. Later, layers further specialize, so that a particular kind of tree is recognized (*Layer 4c*: palm trees) and details are focused (*Layer 4d*: snakes; *Layer 4e*: sombreros). After a further pooling step, layers start to deal with semantic concepts, such as in *Layer 5a* with the detection of the ball. *Layer 5b* then combines semantic concepts of previous layers so that collages can be constructed. Here, one can still identify individual objects, and neurons do not represent isolated meaningful semantic ideas anymore. While considering feature visualizations jointly, an understanding about the meaning of individual neurons can be established.

The challenge to generate meaningful and clear images as they have been presented in the example is faced by the following kinds of feature visualization approaches.

Activation-oriented Visualizations: These are the first kinds of feature visualization approaches to consider gradient-based techniques to generate visualizations

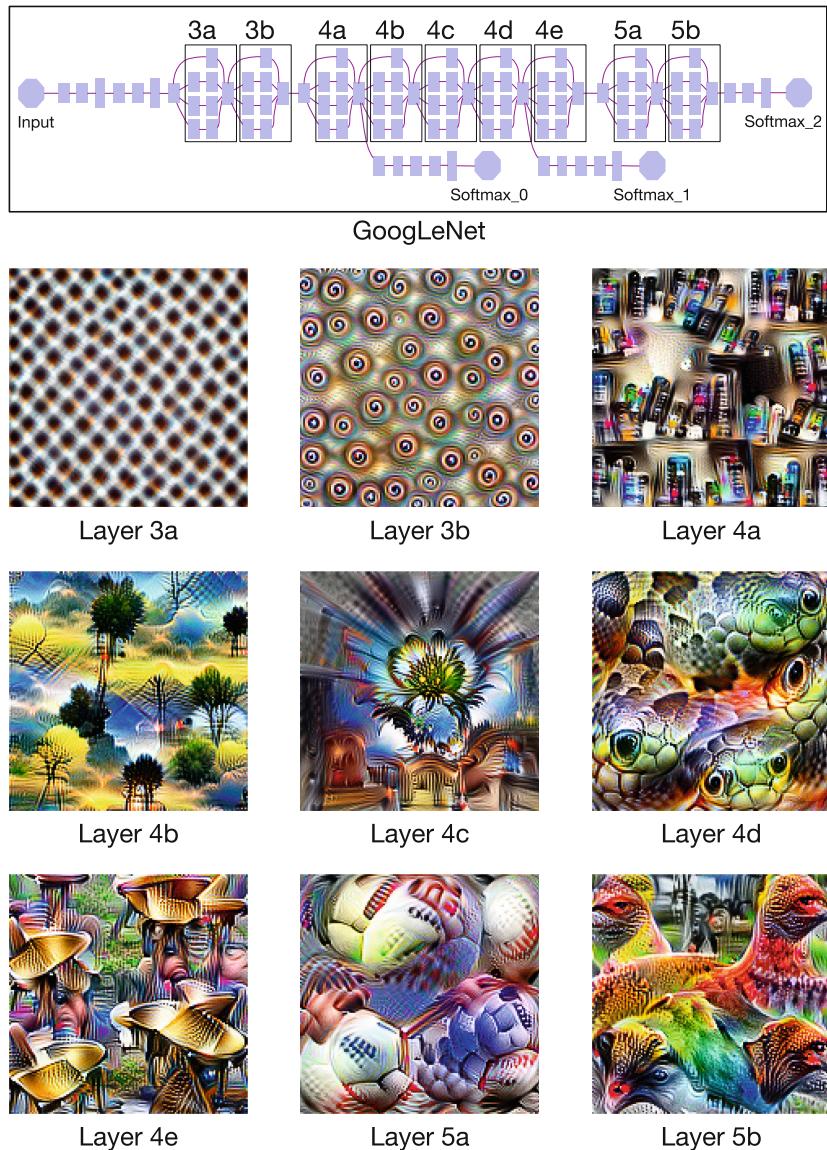


Figure 2.57 Example of a Feature Visualization with GoogLeNet

maximally activating a specific neuron so that the features the neuron learns during the training are revealed.

The *Activation Maximization* approach realizes this as it starts from a random image and updates the image during ascent iterations (Erhan et al., 2009). When the neuron, whose activation is to be maximized, is placed at the output layer and classifies the type of input image, it results in *Class Model Visualization* (Simonyan, Vedaldi, and Zisserman, 2014). Here, an image is generated that maximizes the class score with respect to the input image and the notion of the specific class of an output neuron becomes apparent.

Alternatively, the *Guided Backpropagation* approach builds on the standard back-propagation approach (section 2.5.6.1) but it adds an additional guiding signal at the ReLU layer of Convolutional networks (section 2.5.8.4) so that the backward flow of the positive gradient is kept (Springenberg et al., 2015a). By doing so, generated feature visualizations tend to look sharper.

Regulation-oriented Visualizations: Second kinds of feature visualization approaches draw attention to different kinds of regularization. By maximally penalizing unintended effects at the image generation of a specific neuron, clearer images are intended to be generated. To date, one can find three families of regularization (Olah, Mordvintsev, and Schubert, 2017).

The first family of approaches directly penalizes high frequency noise. The *Feature Inversion* approach focuses on the encoded feature vector that occurs at a certain neuron or part of the network when fed with an input image. This is used to find an image perfectly fitting to this feature vector which will then be used as prior regularization for input images. So, the variance between neighboring pixels is penalized and generated feature visualizations are intended to look more natural (Mahendran and Vedaldi, 2015). Alternatively, one can penalize high-frequency noise by blurring the image with each optimization step (Nguyen, Yosinski, and Clune, 2015).

The second family of approaches directly penalizes encoded feature vectors that do not stand transformations. These include jittering, rotating, or scaling of the image (Mordvintsev, Olah, and Tyka, 2015; Oygard, 2015; Mordvintsev, 2015).

The third family of approaches intend to enforce the learning of the real data. One possibility for this refers to the use of a GAN generator which trains two models simultaneously: a generative model G to capture the data distribution, and a discriminative model D to estimate the probability that a image generated came from the training data rather than G (Nguyen et al., 2016c). Alternatively, one can learn a kind of prior knowledge that gives access to the gradient of probability. Examples can be found at Gaussian Mixture Models being a prior for classes (Mordvintsev, Olah, and Tyka, 2015) and a denoising autoencoder prior to the creation of a generative

model (Nguyen et al., 2016b) or the distance characterization between patches (Wei et al., 2015).

Interim Conclusion: The effect of neurons interacting during the course of activation, which stands aside of dynamic activations, is not issued so far. Here, a modeling of the course of activation is attractive, since the behavioral perspective of the process of neuronal activation becomes explicable.

Further, feature visualization does not issue the meaning of individual neurons as it visualizes rather a static state of a joint activation. Although it is discussed, in how far the generation of diverse images can be supported (Olah, Mordvintsev, and Schubert, 2017; Olah et al., 2018), a coherent concept is still missing. Here, a modeling of the course of activation is ideal which considers neurons as individual processes such that individual effects can be focused and a interpretable neuron specialization can be supported.

2.5.7.3 Saliency

The machine learning community has developed powerful methods that allow the investigation of parts of the input a neuron or a group of neurons of an ANN are looking at (Yu and Shi, 2018) so that the ANN constructor is able to recognize those input signals that caused the most output of a neuron (Zeiler and Fergus, 2013; Simonyan, Vedaldi, and Zisserman, 2014; Springenberg et al., 2015b; Selvaraju et al., 2016; Fong and Vedaldi, 2017; Kindermans et al., 2017b; Sundararajan, Taly, and Yan, 2017; as well as Kindermans et al., 2018). This assignment is the reason behind referring to the same as *saliency*. Some further refer to *attribution*, *relevance*, or *contribution* (Bach et al., 2015; Kindermans et al., 2017b; Olah, Mordvintsev, and Schubert, 2017; as well as Sundararajan, Taly, and Yan, 2017).

On behalf of figures from (Yu and Shi, 2018 and Selvaraju et al., 2016 under CC BY-NC-ND 4.0), Fig. 2.58 intends to clarify the process of saliency visualization with the aid of a collection of different attribution approaches.

While the original image shows both a dog and a cat, four different attribution methods highlight pixels being relevant for the identification of the class representing a cat. These refer to first, *Guided Backprop* (Springenberg et al., 2015a), second, *Grad-CAM* (Selvaraju et al., 2016), third, *Guided Grad-CAM* (Selvaraju et al., 2016), and fourth, *Occlusion Map* (Zeiler and Fergus, 2014). Faced with these approaches, one can see different visualizations results all following the same intention. While different saliency methods are presented in the following sub-section, the requirements for saliency methods are first presented.

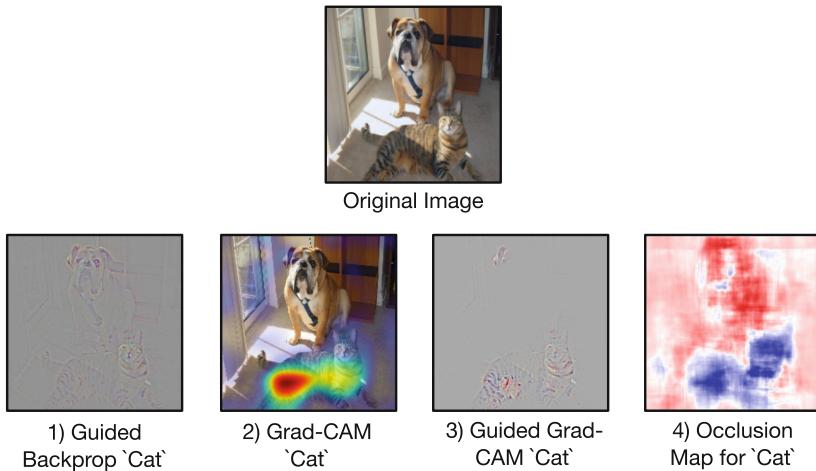


Figure 2.58 Example of Methods Presenting Attribution

2.5.7.3.1 Requirements for Saliency

In order to create adequate saliency visualizations, a collection of requirements can be found in existing literature.

Saliency approaches need to disregard the concrete implementation of ANN, which is denominated as *implementation invariance* (Bach et al., 2015; Kindermans et al., 2017b; Sundararajan, Taly, and Yan, 2017). This considers two ANN as functionally equivalent if their outputs are equal for all inputs.

Approaches need to satisfy *sensitivity* for every input signal. If an input signal differs from a baseline in one feature but leads to different results at a neuron, then the differing feature should be given a non-zero attribution (Bach et al., 2015; Kindermans et al., 2017b; Sundararajan, Taly, and Yan, 2017).

Approaches need to mirror the sensitivity of the ANN model with regard to transformations of the input, such as jittering, rotating, or scaling of the image (Kindermans et al., 2017b), referred to as *input invariance*.

Some demand that attribution approaches satisfy the intuition that linear combined ANN have the same underlying *linearity*. This requirement, for instance, dismisses gradient-based approaches and leads to a weighted sum of its attributes (Sundararajan, Taly, and Yan, 2017).

Further, attributions that consider the difference between an ANN output and the input with regard to its baseline are required (Bach et al., 2015; Kindermans et al., 2017b; Sundararajan, Taly, and Yan, 2017). This shall guarantee that saliency approaches are somewhat comprehensive and is referred to as *completeness*.

2.5.7.3.2 Saliency Approaches

In the literature, one can find three categories of saliency methods (Kindermans et al., 2017b) that intend to build visualizations and present different information about the ANN (Kindermans et al., 2018):

Function-oriented Visualizations: Function methods consider the operations an ANN uses to generate an ANN output when perceiving an input (Kindermans et al., 2018). This goes along with a description of nested functions so that functions on any nesting depth are explained in input space and how the ANN output is extracted from its input can be shown. As deep ANNs are highly nonlinear, functions must be approximated. Instances in literature can be found in the *saliency map*. This visualizes an estimate of how moving along a particular direction in input space influences the ANN output, and the direction is derived from the gradient (Baehrens et al., 2010; Simonyan, Vedaldi, and Zisserman, 2014). A *sensitivity map* visualizes the derivative of the score of the class with regard to the input image (Simonyan, Vedaldi, and Zisserman, 2014). Further, the *guided backpropagation technique* can be applied on the saliency map (Springenberg et al., 2015b), which adds a guidance signal and sharpens the visualization produced. Alternatively, one can add Gaussian noise pixel-wise so that resulting gradients are averaged (Smilkov et al., 2017).

Signal-oriented Visualizations: Signal methods consider the component of data that lead to an ANN activation (Kindermans et al., 2018). As these activations are mapped back to the input space of the ANN, it becomes apparent what input pattern caused a given activation in the feature maps (Zeiler and Fergus, 2014). This goes along with a description of signals, which must be approximated since these correspond to a superposition of what are assumed to be the signal directions of this neuron. Explanations of the same can be found in Kindermans et al. (2017a) and Zeiler and Fergus (2014). Attempts to visualize the signal for deep ANNs further can be found in *DeConvNet* (Zeiler and Fergus, 2014), *Guided BackProp* (Springenberg et al., 2015b), and *PatternNet* (Kindermans et al., 2018). First attempts to consider signals as knowledge generated because of an activation can be found in Gronau and Grum (2017) as well as Grum and Gronau (2018a).

Attribution-oriented Visualizations: Attribution methods focus on the extent the signal dimensions contribute to the output through selected layers (Kindermans et al., 2018). For linear models, this builds on an element-wise multiplication (Kindermans et al., 2018). More complex models build on a decomposition of pixel-wise contributions or the so-called *layer-wise relevance propagation* (LRP) (Bach et al., 2015). Building on this idea, a deep Taylor decomposition (DTD) can be applied in addition (Montavon et al., 2017). Further work on the same can be found in Sundararajan, Taly, and Yan (2017). According to Yu and Shi (2018), one further can find probability-based attribution algorithms, such as those authored by (Zeiler and Fergus, 2014; Zintgraf et al., 2017; as well as Lengerich et al., 2017). With the aid of a visualization called *class activation maps* (CAM), global average pooling (GAP) is used in CNN (Lin, Chen, and Yan, 2013) and discriminative image regions can be identified that are then used by the CNN to identify categories (Zhou et al., 2016). Building on this, the attempt to generalize CAM refers to the *Gradient-weighted Class Activation Mapping* (Grad-CAM) by Selvaraju et al. (2016).

Interim Conclusion: Following Gronau and Grum (2017) as well as Grum and Gronau (2018a), signal-oriented methods are particularly ideal for the CoNM construction as these consider knowledge as ANN activation. Regrettably, a knowledge generation on an ANN level has not been technically constructed till date.

Further, function-oriented methods are ideal for the CoNM construction because these allow the behavioral description of the neuron's meaning for a certain task within the ANN. A modeling of its operations in the sense of processes has not been realized till date although first attempts are available by Gronau and Grum (2017) as well as Grum and Gronau (2018a).

Attribution-oriented visualizations are attractive for the CoNM construction because they issue the contribution of input signals so that, first, the attention for a signal processing can be operationalized and second, the adequateness of learning material is controllable.

2.5.7.4 Neuronal Fuzzy Systems

Since neuronal systems are considered as black boxes because of their bad interpretability, and because prior knowledge cannot be considered transparently, the creation of a hybrid system that connects ANN mechanisms with rule-based systems following fuzzy logics intends to overcome these shortcomings of ANNs with strength of fuzzy systems (Nauck, Klawonn, and Kruse, 1997) in operation with fuzzy sets (Zadeh, 1965).

2.5.7.4.1 Fuzzy Sets

As the human being does not establish mathematical models in everyday life by intuition, heuristic tendencies rather than differential equations are considered (Nauck et al., 2013, p. 151). This can refer to the heuristic example of riding a bike: if a bike's handlebars are moved slightly to the left and the rider leans to the left, the bicycle will turn left.

For the construction of the CoNM, these kinds of vague descriptions are helpful in two ways: first, at the denomination of knowledge generated by ANN so that complex relations can be simplified; and second, at the externalization of a priori knowledge for the ANN design so that simplified human knowledge can be used during the course of learning.

Vague knowledge descriptions are formalized in the theory of fuzzy sets with the aid of a binary view on objects and a *degree of membership*. This issues the question of whether an object is part of a certain set ($= 1$) or not ($= 0$), or if the object can at least partly be associated with this set. Here, a sliding transition from the characteristic to be part of a set to the characteristic not to be part of a set is modeled. Fig. 2.59 visualizes the sliding transition with the aid of the membership function μ .

In this example, from five degrees to eight degrees of the handlebar deflection mentioned and here referred to as x , x satisfies the linguistic term *turn left slightly*. Although a “slight” handlebar movement is not perceived when the handlebar moved left less than three degrees or more than ten degrees because it is either too small or too big, in between these bipolar perspectives, transitions can be identified. Here, the handlebar deflections of four or nine degrees both satisfy the linguistic term *turn left slightly* about a degree of membership of 0.5.

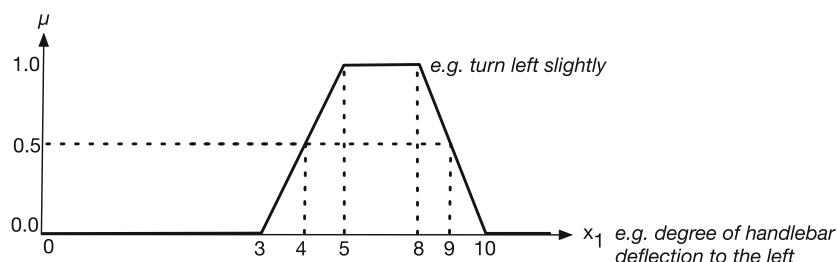


Figure 2.59 Fuzzy Set Example for the Linguistic Term *turn left slightly*

Based on this interpretation, fuzzy sets can be defined as follows (Zadeh, 1965; Lämmel and Cleve, 2012, p. 102; as well as Nauck et al., 2013, p. 154):

Definition 25 (Fuzzy Set).

A fuzzy set over a reference set X is a mapping $\mu : X \rightarrow [0, 1]$ of X in the unit interval, where the value $\mu(x) \in [0, 1]$ is referred to as degree of membership of an element $x \in X$ of the fuzzy set μ .

Here, the process of finding a linguistic term representing fuzzy knowledge about an originally technical signal is referred to as *fuzzyfication*. When fuzzy representations have been finalized, the process of accumulating fuzzy values to non-fuzzy values, for instance, on behalf of the *Center-of-Area Method* or the *Mean-of-Maxima Method* (Nauck et al., 2013, p. 166), is referred to as *defuzzyfication* (Lämmel and Cleve, 2012, p. 110).

2.5.7.4.2 Fuzzy Regulators

Following the intuition of Boolean logic, basic operations are specified as fuzzy logic that enable the construction and deconstruction of fuzzy knowledge sets. Particularly, as the process of knowledge generation or intentional forgetting are intended to be considered with the aid of fuzzy sets, fuzzy operations are ideal for the CoNM construction. Here, one can find operations such as *average*, *union*, *complement*, *conjunction*, *disjunction*, and *negation* (Nauck et al., 2013, p. 159).

Using these operations, models about the reality can be designed that allow the application of fuzzy knowledge within a certain process context. These are referred to as fuzzy systems and can be defined as follows (Nauck et al., 2013, p. 162):

Definition 26 (Fuzzy Systems).

Fuzzy systems $F_{\mathcal{R}}$ are a mapping $F_{\mathcal{R}} : X \rightarrow Y$, in which $X = X_1 \times \dots \times X_n \subseteq \mathbb{R}^n$ is referred to as domain or input space and $Y = Y_1 \times \dots \times Y_m \subseteq \mathbb{R}^m$ is referred to as codomain or output space. These provide the input vector $\underline{x} = (x_1, \dots, x_n) \in X$ and the output vector $\underline{y} = (y_1, \dots, y_m) \in Y$, and \mathcal{R} represents the fuzzy rule base determining the structure of the specific fuzzy system.

As this definition speaks from a *fuzzy rule base*, which is the key element for a fuzzy knowledge consideration, this rule base is further defined by the following:

Definition 27 (Fuzzy Rule Base).

The fuzzy rule base \mathcal{R} consists a set of fuzzy rules $\mathcal{R} = \{R_1, \dots, R_r\}$, in which each rule $R_k \in \mathcal{R}$ is a tuple of fuzzy sets $R_k = (\mu_k^{(1)}, \dots, \mu_k^{(n)}, v_k^{(1)}, \dots, v_k^{(m)})$. Here, $\mu_k^{(i)}$ refers to a fuzzy set of the range of the input variable x_i and $v_k^{(j)}$ refers to a fuzzy set of the range of the output variable y_j .

On behalf of Def. 26 and Def. 27, a concrete definition for fuzzy systems can be established according to Nauck et al. (2013, p. 162):

Definition 28 (Fuzzy System).

A fuzzy system is mathematically modeled by $F_{\mathcal{R}}(\underline{x}) = \underline{y} = (y_1, \dots, y_m)$ with $y_j = \text{defuzz}\left(\bigvee_{R_k \in \mathcal{R}} \{\hat{v}_k^{(j)}\}\right)$ having $\hat{v}_k^{(j)} : Y_j \rightarrow [0, 1]$, $y_j \mapsto \top_2\{\tau_k, v_k^{(j)}\}$ and $\tau_k = \top_1\{\mu_k^{(1)}(x_1), \dots, \mu_k^{(n)}(x_n)\}$, as well as $\text{defuzz} : \mathcal{F} \rightarrow \mathcal{R}$.

Here, \top_1 and \top_2 refer to two kinds of norms, \perp refers to a t-conorm, τ_k refers to the degree of fulfillment of the fuzzy rule R_k , and $\text{defuzz}(\dots)$ models the method for defuzzification. More specifically, \top_1 and \top_2 refer to the t-norm: while the first implements the conjunctive association of the degree of membership by the fulfillment degree of τ_k , the second implements the inference operator for the determination of the consequence of a rule.

2.5.7.4.3 Neuro-Fuzzy Systems

While dealing with fuzzy sets, the idea of neuro-fuzzy systems (NFS) is to determine relevant parameters of a fuzzy system with the aid of procedures from the ANN domain. This might refer to the fine-tuning of parameters or rule building.

As neuronal learning procedures commonly build on gradients which cannot simply be transferred to fuzzy systems, two options are available (Nauck et al., 2013, p. 188). Either, fuzzy systems are adjusted so that they operate with differentiable functions. This is accompanied with a reduced capability to interpret resulting fuzzy ANN structures. Alternatively, ANN learning procedures are adjusted so that they operate in fuzzy structures. This follows a heuristic rather than biologically-inspired proceeding. In order to establish a common understanding for NFS, these are considered to have the following characteristics (Nauck et al., 2013, p. 189):

- A NFS refers to a fuzzy system which is trained with the aid of learning procedures from the ANN domain. Hence, the focus of NFS lies on a data-driven

construction, and the focus of simple fuzzy systems lies on a knowledge-driven construction.

- A NFS may be interpreted as a system of fuzzy rules. These can be initialized with knowledge from the model constructor or from data.
- A NFS considers semantic characteristics of fuzzy systems. This goes along with particular requirements for ANN learning procedures.
- A NFS approximates a n -dimensional function. The fuzzy rules built partial and vague supporting points and can be interpreted as data prototypes.
- A NFS can be considered as a connection-oriented, 3-level-based, feedforward ANN.

Semantics in Fuzzy ANN: The semantic interpretability of an ANN is given by conventional fuzzy systems with the following characteristics (Nauck et al., 2013, p. 263):

- A base of fuzzy rules is available.
- Every rule is allowed to consider various variables for antecedence and consequence.
- The domain of every variable can be described by various linguistic terms.
- Every linguistic term is defined by only one fuzzy set and, accordingly, the membership function.

2.5.7.4.4 Fuzzy ANN Architecture

Some representatives waive architectures for fuzzy ANN (Nauck et al., 2013), and fuzzy ANN techniques can be combined with particular ANN architectures, such as self-organizing maps (Bezdek, Tsao, and Pal, 1992; Vuorimaa, 1994) or fuzzy associative networks (Kosko, 1992), so that they follow the underlying ANN technique's architecture. However, the specific design of fuzzy ANN architectures are determined by the rules to be considered and the learning problem (Berenji, 1992; Nauck and Kruse, 1992; Nauck and Kruse, 1993). Fig. 2.60 intends to clarify an abstracted architecture of NFS which is on behalf of a connection-oriented, 3-level-based representation (Nauck et al., 2013, p. 190). This is exemplified by the XOR example with the following two rules:

R₁: if x_1 is large and x_2 is small then y is large, and
R₂: if x_1 is small and x_2 is large then y is large.

This is such that the data flow within the system and the parallel signal processing is illustrated and the process of fuzzyfication and defuzzification become apparent.

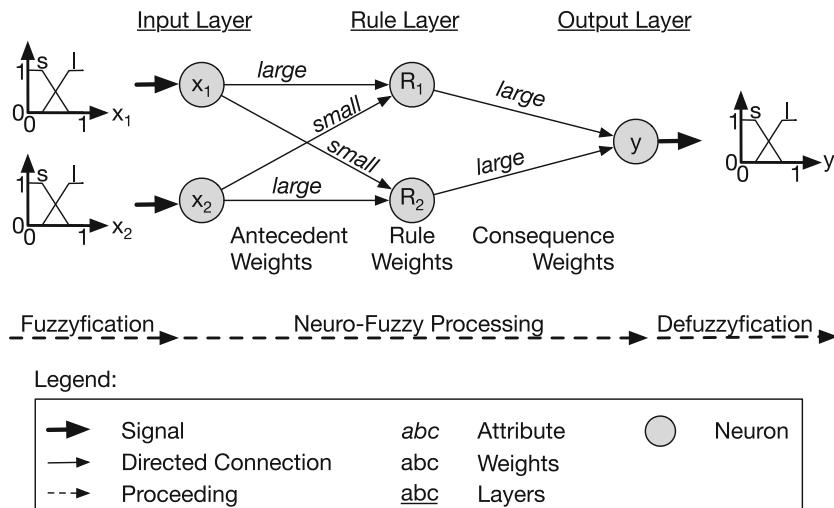


Figure 2.60 Fuzzy ANN Architecture

Here, one can see five levels being represented by three layers as follows: (1) The *input level* corresponds to the input layer of ANN and provides neurons for input variables. In this example, this refers to the two input signals given by the XOR problem. Here, a fuzzyfication is commonly applied. (2) The *antecedent level* can be found in form of weights in between the input layer and hidden layer of ANN. It represents fuzzy sets that are considered as linguistic terms in the antecedent rule component. In this example, this refers to “if x_1 is large and x_2 is small...” (R_1) and “if x_1 is small and x_2 is large...” (R_2). (3) The *rule level* corresponds to the hidden layer of the ANN and represents the fuzzy rules. In this example, for solving the XOR problem, hidden layer neurons provide the *minimum* activation function. (4) The *consequence level* can be found in the form of weights in between the hidden layer and output layer of ANN. It represents fuzzy sets that are considered as linguistic terms in the consequence rule component. In this example, this refers to “...then y is large” (R_1) and “...then y is large” (R_2). (5) The *output level* corresponds to the output layer of ANN and provides neurons for output variables. In this example, for

solving the XOR problem, output layer neurons provide the *maximum* activation function. Here, a defuzzification is commonly applied.

Considering this kind of architecture, a neuronal learning would have to modify parameters of the second and fourth layer (*antecedent level* and *consequence level*). So, it refers to a biologically non-plausible, spatial organization. The storage of membership functions within these two levels guarantees that fuzzy rules are represented by the third level (*rule weights*) only, and further, that these deal with the same linguistic terms. Although these mechanisms are biologically not plausible, they support the interpretability of ANN.

2.5.7.4.5 Fuzzy Learning

The ANN learning approach that is applied on fuzzy ANN must respect the NFS characteristics so that the interpretability of the fuzzy system is not destroyed. Hence, it must provide the following characteristics (Nauck et al., 2013, p. 263):

- Changes of fuzzy sets may not vary the corresponding degree of membership so that values beyond [0, 1] are resulting by the ANN structure.
- The supporter of a fuzzy set (support: $\{x \in X | \mu(x) > 0\}$) must be positioned within the domain of the corresponding variable.
- The ordering of the fuzzy sets of a variable, which is determined by the corresponding supporter, may not be changed.
- Identical linguistic terms may not be represented by various fuzzy sets.
- The appending or changing of rules may not lead to a inconsistent rule base, which means that any rule must pairwise hold different antecedents.

In order to respect these kinds of limitations, either the NFS architecture reflects designs, so that common training algorithms cannot disregard requirements presented, or common training algorithms are modified.

2.5.7.4.6 Interim Conclusion

Principally, NFS are very suitable for the CoNM construction as these support the interpretability of ANN structures built. The principle of NFS enforcing an interpretability by the assumption of architectural components creating semantics is particularly suitable. This includes the adaptation of learning mechanisms and architectural designs.

Despite this allure, upon closer inspection of the techniques, NFSs are not attractive for the sophisticated CoNM construction because of the following challenges and limitations.

Fuzzy systems demand for knowledge about the current learning problem that are expressed in form of linguistic rules. The first challenge refers to the general availability of this kind of knowledge. Therefore, in the CoNM context, NFSs need to, first, consider linguistic rules on behalf of expertise and second, establish a knowledge base on behalf of data so that the explicit use of knowledge is optional.

As linguistic rules are coded within the fuzzy system by design, these rules further need to represent relevant knowledge that is, first, complete, second, correct, and third, consistent. Otherwise, the NFS may not provide plausible solutions (Nauck et al., 2013, p. 183). This leads to a second challenge: since no formal methods are available here, a proceeding is *heuristic* rather than systematic (Nauck et al., 2013, p. 183). Therefore, in the CoNM context, a systematical proceeding must be provided.

When dealing with NFSs, input and output values need to be described by linguistics. As this demands for a binary perspective on the world, information about the extent of changes or dynamic effects are lost for the increase of interpretability. In CoNM contexts, this is not adequate, per se, as machines commonly are dealing with real values and these will lead to more versatile ANN results than the NFS can realize. Therefore, in the CoNM context, versatile values must be considerable which, so far, principally dismiss fuzzyfication.

As a compromise for the limited binary view on the world, one can establish α slices over the underlying basic set X (Nauck et al., 2013, p. 156) so that various *vertical representations* can stand as an approximation as this complicates the process of interpretation. Till date, corresponding mechanisms are still missing. This includes the transparent denomination of any kind of representations such as the process modeling language used to deal with modeling objects.

Since NFSs and non-fuzzy ANNs realize an individual potential, their combination must be enabled. This supports the implementation of individual learning problems and their harmonized integration. Ideally, ANNs are transformable to NFSs and vice versa.

Further, fuzzy ANN demand for uniqueness of linguistic terms. Beside the fact that this is not plausible from a biological point of view, this is not adequate for economic contexts either. An example can be found in the redundant design of business process models where, because of the redundant use of certain process steps, a breakdown will not cause a total process stop.

Although fuzzyfication simplifies the interpretation of precise values, it does not simplify the use of complex patterns. These refer to non-bidirectional views, dynamic behavior, and non-linear relations (Gronau and Grum, 2017) which cannot be considered by fuzzy sets by definition. This also includes the common denomina-

tion possibility of these kinds of dynamic patterns (required by Grum and Gronau, 2018a) and a subsequent research gap becomes apparent.

2.5.8 Network Architectures

Beside traditional network structures presented in section 2.5.5, special architectures have been evolved that each realize individual characteristics. The extent to which these are suitable to represent actual entities is considered in the following sections. Their demand for algorithmic modifications compared to neural networks is also presented.

Aside from prominent network representatives (Deru and Ndiaye, 2019) such as LeNet, AlexNet, VGGNet, YOLO, and PoseNet, the following draws attention to network archetypes as the CoNM will deal with ANNs on a conceptional level, which will also enable the takeover of network representatives.

2.5.8.1 Radial Basis Networks

A *radial basis function network* (RBF) resembles the structure of a three-layer MLP as follows (Kruse et al., 2015, p.81): First, it has an input layer, which is fully connected with its hidden layer neurons. Second, it has a hidden layer, whose neurons provide a *radial function*. This is the reason why every neuron of this layer has an individual *catchment area* with the name of the network architecture it comes from. Third, it has an output layer, whose neurons are fully connected with hidden layer neurons. Because of the particularity of the radial function, the signal transmission differs from MLPs and is described in the following.

Network Input: The incoming activation a_h of a neuron h of the hidden layer is determined by the distance between any input signal $\underline{\mathbf{x}}_i$ and weight $\underline{\mathbf{w}}_{ih}$ from an input neuron i to that hidden layer neuron with the aid of the $dist(\dots)$ function:

$$a_h = dist(\underline{\mathbf{w}}_{ih}, \underline{\mathbf{x}}_i). \quad (2.68)$$

For the determination of distance, different kinds of a measurement can be applied such as the *Mahalanobis* (Mahalanobis, 1936) or the *Minkowski* distance. For the construction of the CoNM, the Minkowski distance is focused on as it is very intuitive:

$$dist_h(\underline{\mathbf{w}}, \underline{\mathbf{x}}) = \left(\sum_{i=1}^n (w_i - x_i)^k \right)^{\frac{1}{k}}. \quad (2.69)$$

Here, the most common special cases refer to $k = 1$ for the Manhattan distance, sometimes called City Block distance, $k = 2$ for the Euclidean distance, and $k \rightarrow \infty$ for the Maximum distance, which means $d_\infty(\underline{\mathbf{x}}, \underline{\mathbf{y}}) = \max_{i=1}^n |x_i - y_i|$ (Kruse et al., 2015, p.82).

The neuron's catchment area is further specified by the application of an activation function which resembles the activation of hidden layers of MLPs (Eq. 2.35):

$$b_h = \theta_h(a_h). \quad (2.70)$$

The difference in RBFs is that $\theta(\dots)$ refers to a radial activation function which is monotonously decreasing:

$$\theta : \mathbb{R}_0^+ \rightarrow [0, 1] \text{ with } \theta(0) = 1 \text{ and } \lim_{a \rightarrow \infty} \theta(a) = 0. \quad (2.71)$$

Considering the concrete distance measurement of Eq. 2.68 and the concrete activation function of Eq. 2.70, the following interpretation can be established: weights from the input layer neuron to an hidden layer neuron represent the geometrical center of the catchment area because the distance is determined between the weight vector and the input vector. The form of the catchment area is characterized by the distance function $dist(\dots)$. The size of the catchment area is determined by the radius σ , that starts at the catchment area's center and is limited in a strict manner as the activation function specifies.

Here, it is important to remark that the catchment area mentioned does not equal local perception fields or the spatial positioning of neurons within the ANN similar to CNN (section 2.5.8.4), Kohonen maps and SOMs (section 2.5.8.5.1), or neuronal gas (section 2.5.8.5.2). Rather, they refer to a geometrical interpretation.

Activation Functions: For the application of radial activation functions to any hidden layer neuron defined in Eq. 2.70, representatives holding the condition of Eq. 2.71 refer to the following examples (Kruse et al., 2015, p.83).

The *Square Function* realizes an activation if the incoming activation signal falls below the threshold σ :

$$\theta(a) = \begin{cases} 0, & \text{if } a > \sigma \\ 1, & \text{else.} \end{cases} \quad (2.72)$$

$$\frac{\partial \theta(a)}{\partial a} = 0. \quad (2.73)$$

The *Triangle Function* realizes a linearly decreasing activation up to the threshold σ :

$$\theta(a) = \begin{cases} 0, & \text{if } a > \sigma \\ 1 + \frac{a}{\sigma}, & \text{else} \end{cases} \quad (2.74)$$

$$\frac{\partial \theta(a)}{\partial a} = \begin{cases} 0, & \text{if } a > \sigma \\ -\frac{1}{\sigma}, & \text{else.} \end{cases} \quad (2.75)$$

The *Cosine Until Zero Function* realizes a soft decreasing activation up to the threshold 2σ :

$$\theta(a) = \begin{cases} 0, & \text{if } a > 2\sigma \\ \frac{\cos(\frac{\pi}{2\sigma}a) + 1}{2}, & \text{else} \end{cases} \quad (2.76)$$

$$\frac{\partial \theta(a)}{\partial a} = \begin{cases} 0, & \text{if } a > 2\sigma \\ -\frac{\pi \sin(\frac{\pi}{2\sigma}a)}{4\sigma}, & \text{else.} \end{cases} \quad (2.77)$$

The *Gaussian Function* realizes an alternative soft decreasing activation up to the threshold 2σ :

$$\theta(a) = e^{-\frac{a^2}{2\sigma^2}} \quad (2.78)$$

$$\frac{\partial \theta(a)}{\partial a} = \frac{-ae^{-\frac{a^2}{2\sigma^2}}}{\sigma^2}. \quad (2.79)$$

Network Output: The signal transmission of neurons being placed at the output layer is derived equally to MLPs with Eq. 2.34 and Eq. 2.34. Separate from MLPs, output neurons provide a linear transfer function $\theta(\dots)$ with a bias value θ :

$$\theta(a) = a - \theta \quad (2.80)$$

$$\frac{\partial \theta(a)}{\partial a} = 1. \quad (2.81)$$

Variants to linear output activations can consider logistic activation functions too, although this must be circumstantially considered in RBF initialization (Kruse et al., 2015, p. 83).

Training Variants: Training of RBFs can be carried out in many different ways. The process of adjusting the parameters is usually treated as in a typical neuronal network (Schwenker, Kestler, and Palm, 2001, p. 441). Alternatively, phase-wise learning procedures have been established as follows:

In a *one-phase learning* procedure, only output layer weights are adjusted in fixed learning tasks. Here, RBF centers are sub-sampled from the set of input vectors and RBF width are set equal to a predefined real number.

In a *two-phase learning* procedure, the hidden layer and the output layer are trained separately. In a first step, RBF centers and scaling parameters are identified. Cluster centers can either be randomly sampled or obtained by free learning tasks with the aid of clustering approaches *k-means* (Moody and Darken, 1989), supervised learning vector quantization *LVQ* (Schwenker et al., 1994), and supervised training of decision trees *D-Tree* (Kubat, 1998; Schwenker and Dietrich, 2000). In a second step, output weights are adjusted by fixed learning tasks such as the *gradient descent* (Bishop, 1995) or *pseudo inverse solution* approaches (Hertz, Palmer, and Krogh, 1991).

The *three-phase learning* procedure builds on the first two steps of a two-phase learning and in a third step, all RBF parameters are fine-tuned with the aid of backpropagation algorithms (section 2.5.6). Here, for the full parameter set, $\underline{\mathbf{U}} = (\underline{\mathbf{c}_j}, \sigma_j, \underline{\mathbf{w}_j})$ is moved the small distance α in the direction of the negative error gradient $-\nabla E(\dots)$ as follows (Schwenker, Kestler, and Palm, 2001, p. 445):

$$\underline{\mathbf{U}}^{t+1} = \underline{\mathbf{U}}^t - \alpha \nabla E(\underline{\mathbf{U}}^t). \quad (2.82)$$

Assuming we have a RBF with Gaussian radial functions (Eq. 2.78) and the Minkowski distance with $k = 2$ (Eq. 2.69), depending on the choice of the error consideration (batch or online learning defined in section 2.5.6.1.2), we have the following for the weight adjustment between hidden layer neurons j and output neurons k :

$$\Delta w_{j,k} = \alpha \sum_{i=1}^n \theta_j(\text{dist}(\underline{\mathbf{w}}_{ij}, \underline{\mathbf{x}}_i))(t_k - b_k). \quad (2.83)$$

For the center adjustment, which refers to the weights from input neurons i to hidden layer neurons j , we have the following:

$$\Delta c_{i,j} = \alpha \sum_{i=1}^n \theta_j(\text{dist}(\underline{\mathbf{w}}_{ij}, \underline{\mathbf{x}}_i)) \frac{x_i - c_{i,j}}{\sigma_{i,j}^2} \sum_{p=1}^L w_{j,p}(t_p - b_p). \quad (2.84)$$

And for the RBF width, we have

$$\Delta \sigma_{i,j} = \alpha \sum_{i=1}^n \theta_j(\text{dist}(\underline{\mathbf{w}}_{ij}, \underline{\mathbf{x}}_i)) \frac{x_i - c_{i,j}}{\sigma_{i,j}^3} \sum_{p=1}^L w_{j,p}(t_p - b_p). \quad (2.85)$$

Although many applications suggest a separated training of the RBF's layers, and various phases for learning procedures have even been established, these lead to results that are worse than MLPs (Michie, Spiegelhalter, and Taylor, 1994). Since a training of the whole network leads to a better performance than MLPs (Poggio and Girosi, 1990), for the construction of the CoNM, a joint training is ideal as has been drawn in section 2.5.6 and considering Eq. 2.83–2.85.

2.5.8.2 Partially Recurrent Networks

Aiming to consider history by ANNs, partially recurrent networks are a version of recurrent networks presented in section 2.5.5.3. Depending on the learning task, an adequate *input window* can be designed so that the neuronal processing depends on former activations by architecture. Here, this refers to *partially recurrent networks* (Lämmel and Cleve, 2012) since recurrent connections are not established throughout. The following paragraphs present prominent research that distinguish the same from fully recurrent connected networks as shown in Fig. 2.52.

2.5.8.2.1 Jordan Networks

Jordan networks consider history by a feedback of the network's output with the aid of *context neurons* whose number refers to the number of output layer neurons. Further neurons are fully connected such as neurons of the input layer connected with hidden layer neurons and the latter kind of neurons with output layer neurons. Fig. 2.61 visualizes the foundational architecture.

The algorithmic anomaly lies in the following: neurons of the output layer are connected by a one by one connection to context neurons. Since the only backward directed kinds of connections (from output layer neurons to context neurons) are set to one and the meaning of context neurons is considered by adaptable

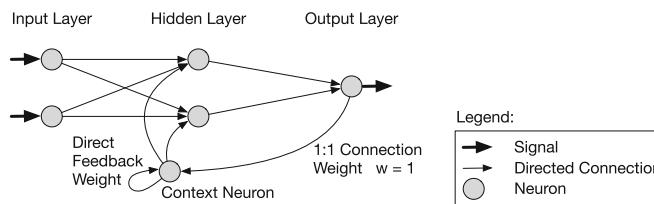


Figure 2.61 Jordan Network Architecture

connections from context neurons to hidden layer neurons, feedback connections are not part of the training process. Hence, a training only needs to consider feedforward connections, and a simple backpropagation can be applied (section 2.5.6.1).

2.5.8.2.2 Elman Networks

Elman networks consider history by a feedback of the network's hidden layer with the aid of *context neurons* whose number refers to the number of hidden layer neurons. Further neurons are fully connected such as neurons of the input layer connected with hidden layer neurons and the latter kind of neurons with output layer neurons. Fig. 2.62 visualizes the foundational architecture.

The algorithmic anomaly lies in the following: neurons of the hidden layer are connected by a one by one connection to context neurons. Since the only backward directed kinds of connections (from hidden layer neurons to context neurons) are set to one and the meaning of context neurons is considered by adaptable connections from context neurons to hidden layer neurons, feedback connections are not part of the training process. Hence, a training only needs to consider feedforward connections, and a simple backpropagation can be applied (section 2.5.6.1).

As the size of the hidden layer is not dependent on the learning task, the Elman networks tend to be more powerful than Jordan networks (paragraph 2.5.8.2.1). The size of the hidden layer and the number of context neurons can simply be increased such that the Elman networks tend to produce better results than the Jordan networks (Lämmel and Cleve, 2012, p. 249).

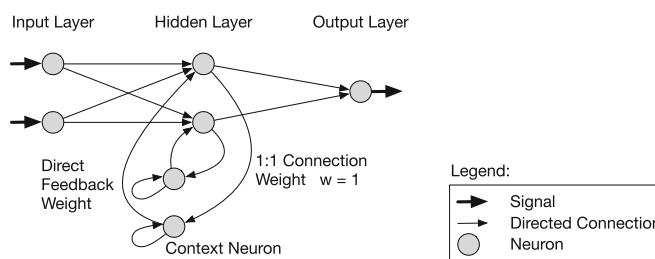


Figure 2.62 Elman Network Architecture

2.5.8.3 Long-Short-Term-Memory Blocks

In the following section, *long-short-term memory* (LSTM) blocks are explained with their working behavior as isolated memory blocks. They are considered as foundational to the CoNM because of their ability to overcome a weakness of recurrent networks as they were defined in 2.5.5.3. Here, errors flow back in time and either blow up or vanish. This is called the *Vanishing Gradient Problem* (Hochreiter, 1991; Bengio, Simard, and Frasconi, 1994; Hochreiter et al., 2001; Graves, 2012). In the first case, this error leads to oscillating weights since the backpropagated error depends exponentially on the network's weights. In the second case, the structure does not learn bridging long times at all, or does so very slowly, since the backpropagated errors are so small. This is why the memory of ordinary recurrent networks is called short-term memory. LSTM blocks overcome this problem by introducing three multiplicative neurons per hidden neuron called gate keepers. These ensure to keep a cell's relevant information over a long period of time and intend to establish a long-term memory.

Based on Graves (2012), Fig. 2.63 tries to visualize the Vanishing Gradient Problem and show the sensitivity of nodes to the input over time. A dark blue shade represents a great sensitivity and a bright blue shade represents a low sensitivity.

In (a), one can see that the sensitivity of recurrent neural networks decays exponentially over time: because of new input activation at every new time step, the activation of a hidden unit will be overwritten. The shading of a hidden neuron becomes brighter over time. One can say that the network forgets the information of previously-seen data exponentially over time and only short-term memory is addressed. As (Hochreiter et al., 2001) discovered, it is hard for the network to learn relations that have delays greater than 10 time steps.

In (b), one can see that a LSTM block within the hidden layer preserves the gradient information over time. This is accomplished using three gate keepers called

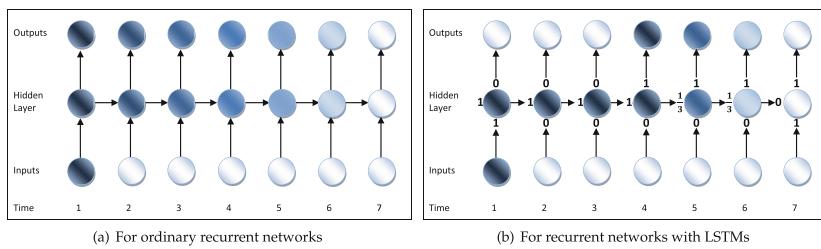


Figure 2.63 Vanishing Gradient Problem

the *input gate*, *forget gate*, and *output gate*. Their detailed explanation can be found in section 2.5.8.3.1. Each of their time-dependent states is placed in the picture on the bottom, on the left, and on the top of the hidden neuron. Since the input gate only lets an activation pass in the first time step and the forget gate lets the activation pass completely until the fourth time step, the memory cell *remembers* the first activation without any loss. An exponential information loss and even an entire *black out* can be learned as well at the hidden neurons of the time step 4 – 5 and 6. The sensitivity of the output units can be controlled by the output gate without affecting the cell itself. Of course, the working behavior of the gate keepers has to be learned as well. When working correctly, LSTM blocks can easily learn relations that have delays greater than 1.000 time steps (Hochreiter and Schmidhuber, 1997a). This is why LSTM blocks add a form of long-term memory to recurrent neural networks and the memory block itself is called an LSTM block.

The following subsections explain this procedure. Their special architecture will be explained in 2.5.8.3.1, then the computation of LSTM networks will be shown in section 2.5.8.3.2 as it will be used in the thesis's experiments as well. Some LSTM block specific learning proceedings will be explained in section 2.5.8.3.3.

Although a bidirectional version of LSTM blocks is available (Graves and Schmidhuber, 2005) that considers the past and the future of a current state, the following focuses on unidirectional LSTM blocks only since it is in harmony with the processual understanding of Def. 3: future states are to be predicted and, therefore, may not be part of the recurrent learning context.

2.5.8.3.1 Block Architecture

The starting point for the explanation of the LSTM block architecture shall be a standard recurrent hidden layer unit as it was defined in 2.5.5.3. Here, the activation of previous time steps is overtaken naively via its recurrent connections. In its place, a recurrent LSTM block will be set which can be seen as a recurrent subnet that places three multiplicative neurons around the previously mentioned standard recurrent unit, which is subsequently referred to as a block's *memory cell*.

Apart from some formulas which will be explained in section 2.5.8.3.2, the architecture of LSTM blocks is visualized in Fig. 2.64.

Such multiplicative units are called gate keepers and are visualized in green throughout this thesis. They will influence the standard working of the naive memory cell inside in three ways:

First, it is shielded by an input gate, which decides what kind of activation is important to remember and is kept next to the current activation of the recurrent memory cell itself. The more important the current activation, the more its gate

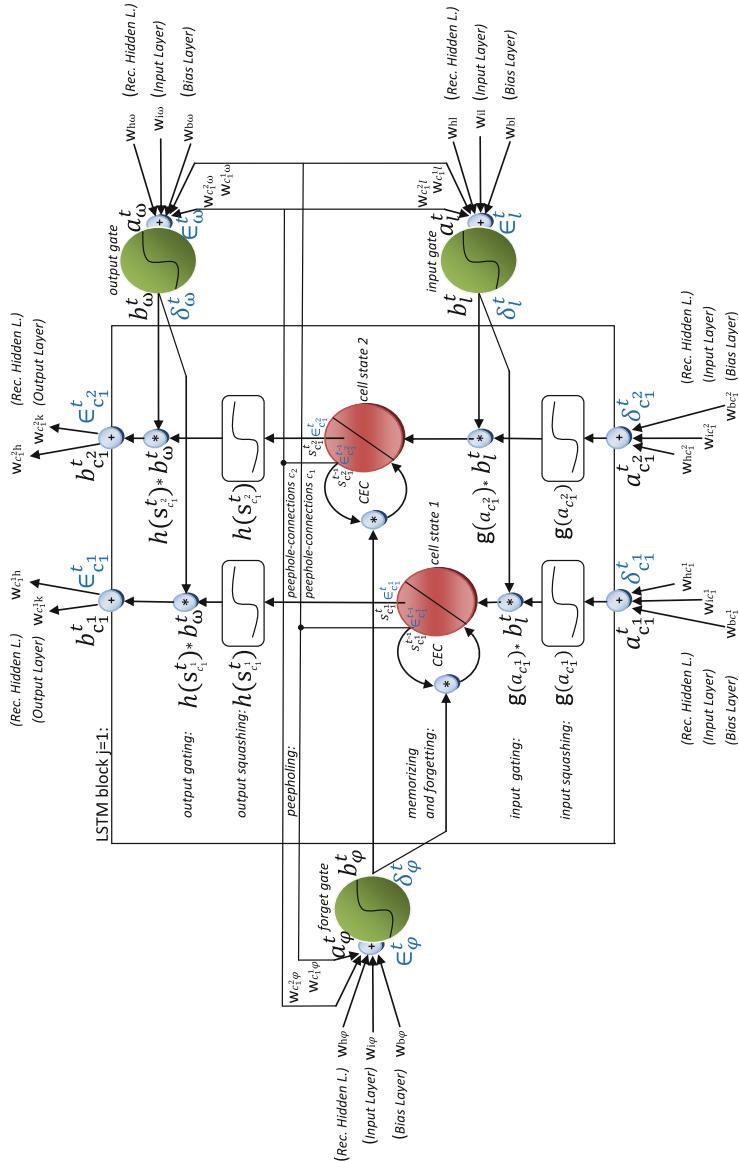


Figure 2.64 LSTM Block Architecture Having Two Memory Cells

will be opened. Second, it is shielded by an output gate, which decides what kind of current memory activation is important to produce a result for the current time step or when to prevent other units by that cell activation. The more important the current memory activation, the more its gate will be opened. The more misleading its activation, the more its gate will be closed. Third, a forget gate decides how much of the current memory activation should be forgotten. The more it opens its gates, the less will be forgotten. In this way, some useful activations can cycle for more than 1000 time steps via the block internal recurrent connections. This won't be used until it is needed.

The activation of the recurrent memory cell itself is used twice: first, it is used for the production of the block output; and second, it is used for the behavior of the gating units. Since the block internal working behavior is invisible for the rest of the network, the gating units are fed (apart from further network connections) with the activation of the blocks internal memory cell. This is why the connections between the memory cell and the gating units are called *peephole connections*.

Since LSTM blocks are designed to be memory blocks, their capacity can be increased by incrementing the number of memory cells within the LSTM block. Every block internal memory cell shares the same three gate keepers, but is connected with the proper peephole connections between them. The more cells a block contains, the greater its capacity. Experiments on the comparison of the capacity of several LSTM blocks containing one cell and the capacity of one LSTM block containing the same number of memory cells are still lacking.

Since the computation of recurrent networks containing LSTM blocks gets much more complex through the specifications of the LSTM block architecture, rather than computations of ordinary recurrent networks defined in 2.5.5.3, it will be treated separately in the following section.

2.5.8.3.2 Calculation

This section deals with the formulas of the activation (forward pass) of a LSTM block within a recurrent neural network and its gradient derivation (backward pass).

Since the CoNM will consider further neurons standing next to LSTM blocks, such as bias neurons for every layer, the following equations define the operation of LSTM blocks as part of recurrent neuronal networks defined in section 2.5.5.3). The following formulas are representing a network having I inputs, K outputs, and having H neuronal elements in a single hidden layer containing LSTM blocks with $C_j = \{c_j^{(n_c)}\}$ cells, where n_c denotes the number of cells within LSTM block j . Since LSTM block can stand next to ordinary neurons, they are not treated separately.

The equations are only given for one single LSTM block j . If there were several, the same have to be applied for each in any order. As before, the weight w_{ih} connects input unit i with hidden unit j , a stands for the unit's input, and b stands for the unit's output after the unit's working is done (in the simplest case of the application of a transfer function). As we are focusing on LSTM blocks, there are four existing kinds of w_{ih} : weights to the cells w_{ic} and weights to the three gating units w_{il} , $w_{i\phi}$, and $w_{i\omega}$. The subscript l represents the input gate of a specific LSTM block, while ϕ represents its forget gate and ω represents its output gate.

For each sequence, one forward pass and the corresponding backward pass will be done. The formulas and relevant nomenclature is visualized in Fig. 2.64.

2.5.8.3.2.1 Forward Pass

The chronology in which the following steps proceed is important and it should be carried out as specified below. While starting with the forward pass calculation at $t = 1$ and incrementing it up to T , states and activations are set to zero at the beginning of each sequence.

Squashing Functions: Given below is the transfer functions $f(\dots)$ which is used at every gating unit, $g(\dots)$ which is used to squash every cell input, $h(\dots)$ which is for output squashing of every cell, and $i(\dots)$ which is used as transfer function for every output unit of the net:

$$f(a) = i(a) = \frac{1}{1 + e^{-a}} \text{ see Eq. 2.20} \quad (2.86)$$

and

$$g(a) = h(a) = \tanh(a) \text{ see Eq. 2.18.} \quad (2.87)$$

Input Gates: The LSTM block's specific input of its input gate a_l at time step t consists of the following parts: the net input is passed through linear neurons $i \in I$ that are connected with the input gate with the weights w_{il} . As we define a recurrent network, the LSTM block might be connected with the other hidden layer neurons $h \in H$, whose activations b_h of the previous time step $t - 1$ are considered. Each LSTM block has to consider the activation of its containing cells $c_j^n \in C_j$ itself, which can be seen as the state of the block s_c . Since the cell state should be protected by the input gate, the state of the previous time step $t - 1$ is taken into consideration. It is connected with the input gate via the peephole connections w_{cl} . Additionally, there might be a block specific bias which is modeled with bias neurons $bias \in Bias$

that used to be singular. Since they are bias neurons, their activation is 1 and their weights w_{bl} stand for the bias itself.

$$a_l^t = \sum_{i=1}^I w_{il}x_i^t + \sum_{h=1}^H w_{hl}b_h^{t-1} + \sum_{c=1}^{C_j} w_{cls_c^{t-1}} + \sum_{b=1}^{Bias_j} w_{bl} \quad (2.88)$$

The block specific input of the input gate is transferred via the transfer function $f(\dots)$ and results in the activation of the input gate of that block at time t .

$$b_l^t = f(a_l^t) \quad (2.89)$$

Forget Gates: In a similar way, the input of the block specific forget gate a_ϕ and its activation b_ϕ at time step t are calculated. The notation is coherent with the input gate.

$$a_\phi^t = \sum_{i=1}^I w_{i\phi}x_i^t + \sum_{h=1}^H w_{h\phi}b_h^{t-1} + \sum_{c=1}^{C_j} w_{c\phi}s_c^{t-1} + \sum_{b=1}^{Bias_j} w_{b\phi} \quad (2.90)$$

$$b_\phi^t = f(a_\phi^t) \quad (2.91)$$

Cell Input & State: Each of the block's cells has a cell specific input a_c of time step t and memorizes its activation within a separate cell state s_c . The cell input is built similarly with respect to the gate inputs. This input will be squashed by the transfer function $g(\dots)$ and filtered by the current input gate activation. The previous state will be filtered by the current forget gate activation and together, they result in the new cell state s_c at time step t .

$$a_c^t = \sum_{i=1}^I w_{ic}x_i^t + \sum_{h=1}^H w_{hc}b_h^{t-1} + \sum_{b=1}^{Bias_j} w_{bc} \quad (2.92)$$

$$s_c^t = b_\phi^t s_c^{t-1} + b_l^t g(a_c^t) \quad (2.93)$$

This context is visualized in Fig. 2.64.

Output Gates: The notation of the output gate is coherent with that of the input gate. The input of the block specific forget gates a_ϕ and its activation b_ϕ at time step t are calculated in a similar way, with the only difference being in the selection of

the cell state s_c . Since the cell activation should be usable at once, the output gate is influenced by the state of the current time step t .

$$a_\omega^t = \sum_{i=1}^I w_{i\omega} x_i^t + \sum_{h=1}^H w_{h\omega} b_h^{t-1} + \sum_{c=1}^{C_j} w_{c\omega} s_c^t + \sum_{b=1}^{Bias_j} w_{b\omega} \quad (2.94)$$

$$b_\omega^t = f(a_\omega^t) \quad (2.95)$$

Cell Outputs: Since every cell has a proper cell state s_c , each produces a separate LSTM block output. For this, the cell state s_c at time step t will be squashed by the transfer function $h(\dots)$ and filtered by the current output gate activation b_ω .

$$b_c^t = b_\omega^t h(s_c^t) \quad (2.96)$$

Net Output The net output layer for the case of Schmidhuber consists of $k \in K$ logistic sigmoid neurons that sum up the outputs b_h^t coming from the previous hidden layer with respect to the transfer function $i(\dots)$, as shown in section 2.5.5.1. Here, one can consider the following three kinds of activations: first, the activation coming from neurons of the hidden layer. Since each of the LSTM blocks can have several cells and with this several cell outputs, each of the cell outputs belongs to the outputs of the hidden layer. Second, the activation coming from the input activation, and third, the activation coming from bias neurons being connected with the output layer neurons.

$$b_k^t = i \left(\sum_{h=1}^H w_{hk} b_h^t + \sum_{i=1}^I w_{ik} x_i^t + \sum_{b=1}^{Bias_{out}} w_{bk} \right) \quad (2.97)$$

2.5.8.3.2.2 Backward Pass

The chronology in which the following steps proceed is important for the backward pass as well. It should be carried out as specified below. While starting with the backward pass calculation at $t = T$ and decrementing it down to 1, δ and ϵ terms are set to zero at $t = T + 1$ of each sequence.

Network Output: When the injection error E^T , as it was defined in 2.5.6.1, is injected at each output neuron $k \in K$ at time t , the output error ϵ_k^t is given by the following equation. As objection function, one of various can be chosen here (cf. section 2.5.6.4). Hence, the formulas for the output layer are kept in a general version in the following:

$$\epsilon_k^t = \frac{\partial E^T}{\partial b_k^t}. \quad (2.98)$$

The corresponding input error δ_k^t can be found with the help of ϵ_k^t :

$$\delta_k^t = \epsilon_k^t \cdot \frac{\partial b_k^t}{\partial a_k^t}. \quad (2.99)$$

As a_k^t can be found in Eq. 2.97, its derivative can be derived for the gradient calculation:

$$\frac{\partial a_k^t}{\partial w_{hk}}. \quad (2.100)$$

With this, the gradient to be minimized with respect to the weights from the output neurons to the LSTM blocks can be derived:

$$\nabla_{hk} = - \sum_{t=1}^T \frac{\partial E^T}{\partial b_k^t} \cdot \frac{\partial b_k^t}{\partial a_k^t} \cdot \frac{\partial a_k^t}{\partial w_{hk}}. \quad (2.101)$$

Cell Outputs: The error coming to a certain LSTM block at time step t is called cell output error ϵ_c . It is dependent on the error coming from the output layer δ_k at time step t and on the error coming from the recurrent connections of the hidden layer called δ_h of the previous time step $t + 1$ as follows:

$$\epsilon_c^t \stackrel{\text{def}}{=} \frac{\partial E^T}{\partial b_c^t} = \sum_{k=1}^K w_{ck} \delta_k^t + \sum_{h=1}^H w_{ch} \delta_h^{t+1}. \quad (2.102)$$

Output Gates: The error leaving the output gate is called δ_ω . It is dependent on the cell output error ϵ_c of that time step t :

$$\delta_\omega^t = \frac{\partial E^T}{\partial b_c^t} \cdot \frac{\partial b_c^t}{\partial a_\omega^t} = \epsilon_c^t \cdot f'(a_\omega^t) \sum_{c=1}^{C_j} h(s_c^t). \quad (2.103)$$

States: The state error ϵ_s of time step t is dependent on five summands: first, the error coming to a certain LSTM block ϵ_c at time step t ; second, the error being taken in the memory itself called ϵ_s of the previous time step $t + 1$; third, the error propagated back via the input gate of the previous time step $t + 1$ called δ_l ; fourth, the error propagated back via the forget gate of the previous time step $t + 1$ called

δ_ϕ ; and fifth, the error propagated back via the output gate of that time step t called δ_ω . This context is visualized in Fig. 2.64 in blue.

$$\epsilon_s^t \stackrel{\text{def}}{=} \frac{\partial E^T}{\partial s_c^t} = \frac{\partial E^T}{\partial b_c^t} \cdot \frac{\partial b_c^t}{\partial s_c^t} + \frac{\partial E^T}{\partial s_c^{t+1}} \cdot \frac{\partial s_c^{t+1}}{\partial s_c^t} + \frac{\partial E^T}{\partial a_c^{t+1}} \cdot \frac{\partial a_c^{t+1}}{\partial s_c^t} + \frac{\partial E^T}{\partial a_\phi^{t+1}} \cdot \frac{\partial a_\phi^{t+1}}{\partial s_c^t} + \frac{\partial E^T}{\partial d_\omega^t} \cdot \frac{\partial d_\omega^t}{\partial s_c^t} \quad (2.104)$$

$$\epsilon_s^t = \epsilon_c^t \cdot b_\omega^t h'(s_c^t) + \epsilon_s^{t+1} \cdot b_\phi^{t+1} + \delta_l^{t+1} \cdot w_{cl} + \delta_\omega^{t+1} \cdot w_{c\phi} + \delta_\omega^t \cdot w_{c\omega} \quad (2.105)$$

Cells: The error leaving a certain LSTM block at time step t is called cell input error δ_c . It is dependent on the error coming from the corresponding cell state ϵ_s of time step t as follows:

$$\delta_c^t = \frac{\partial E^T}{\partial s_c^t} \cdot \frac{\partial s_c^t}{\partial a_c^t} = \epsilon_s^t \cdot b_l^t g'(a_c^t). \quad (2.106)$$

Forget Gates: The error leaving the forget gate is called δ_ϕ . It is dependent on the cell state error ϵ_s of that time step t :

$$\delta_\phi^t = \frac{\partial E^T}{\partial s_c^t} \cdot \frac{\partial s_c^t}{\partial a_\phi^t} = \epsilon_s^t \cdot f'(a_\phi^t) \sum_{c=1}^{C_j} s_c^{t-1}. \quad (2.107)$$

Input Gates: The error leaving the input gate is called δ_l . It is also dependent on the cell state error ϵ_s of time step t :

$$\delta_l^t = \frac{\partial E^T}{\partial s_c^t} \cdot \frac{\partial s_c^t}{\partial a_l^t} = \epsilon_s^t \cdot f'(a_l^t) \sum_{c=1}^{C_j} g(a_c^t). \quad (2.108)$$

The errors defined for each neuron within the LSTM block considered serve to derive the individual gradients by Eq. 2.44 and carry out the learning procedure within the LSTM block.

2.5.8.3.3 Learning with LSTM Blocks

The question of how to determine the size of the network's hidden layer containing LSTM blocks remains. Since the numbers of input and output neurons have to be chosen regarding the problem, the architecture and with this the number of elements

in the hidden layer has to be chosen separately. According to Trowitzsch (2011), who analyses theorems of universal approximation, there is no simple rule to determine how many hidden layer units are enough: too few may limit the network so that it is not able to learn the problem properly, while too many will waste computation time and tend to overfit (cf. section 2.5.6.7). Thus, one can choose from three approaches which vary in the freedom of the selection of that number and the final use of those elements:

1.) Trial-and-Error: Here, one has the freedom to chose only a fixed number of elements in the hidden layer and try another when the selected one fails. Such elements are used for learning all the time.

2.) Sequential Network Construction: Here, the net's size is automatically increased whenever the error stops decreasing. Hence, the old learning results are kept while new learning power is enabled. One does not have to know the necessary number of elements in the previous hidden layer since the algorithm finds it. Further, since elements are added when they are needed, the existing elements are also used for learning all the time. Based on the work of Fahlman (1991), the algorithm has been applied to four variations on LSTM expansions and their performance has been analyzed by Ribeiro and Alquézar (2002).

3) Output gate bias: Here, one has to define a number of elements in the previous hidden, but not all elements are used for learning all the time since all gates get an element specific and negative initial gate bias. The bias pushes the initial memory cell activations to zero until the bias is corrected through learning. From this point on, it can be used for learning. Since every block gets a specific bias, this is as if they were added to the network at different iteration steps. The more negative the initial bias, the later the block participates in learning. Hochreiter and Schmidhuber (1997a) has realized several experiments on the same.

Furthermore, the last two approaches have the advantage of ensuring that the error reduction is realized by storing information over time. While learning normally, all available memory cells might be abused, i.e., when they are misused as a threshold for other units or when two memory cells learn to represent the same information and both participate in the resulting production. The problem of those cells is that they do not use their learning potential and it may take a long time to release them.

For the creation of the CoNM concept, the third learning approach (*output gate biases*) is suitable as the problems mentioned can be avoided.

2.5.8.4 Convolutional Networks

The architecture of Convolutional neuronal networks (CNN) consists of two parts: the detection and the identification parts (Schwaiger and Steinwendner, 2019, p. 187). The *detection part* contains repeated convolutional layers, activation layers, and pooling layers, that jointly extract complex attributes from a feature space, such as images (Deru and Ndiaye, 2019, p. 81). These intend to first recognize simple structures from which later layers can recognize more complex structures so that an input signal is transformed to one detection vector describing the input features. The *identification part* consists of arbitrary complex ANN and can refer to any kind of machine learning approach as well. It considers the codification generated by the detection part and generates a classification for the underlying learning problem. Fig. 2.65 presents an overview of CNN architecture and the following introduces its architectural components.

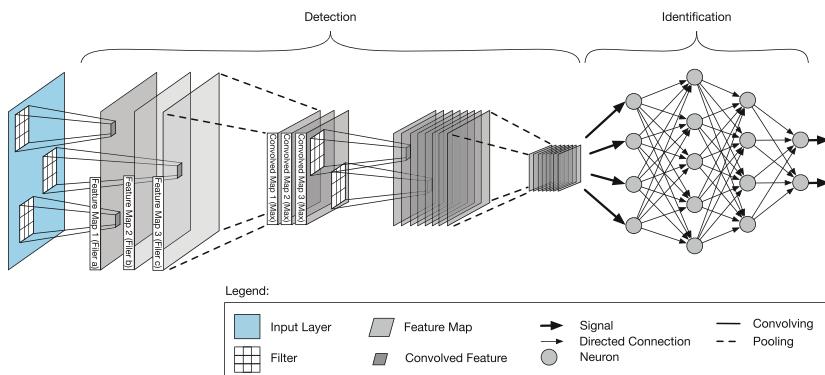


Figure 2.65 Convolutional Neuronal Network Architecture

Convolutional Layers: Each convolutional layer consists of a set of neurons from which each neuron only focuses on a small part of the feature space and reacts only on a certain pattern. This is biologically motivated by neurons having a local perception.

The specific kind of pattern is modeled by a filter that is applied to an input signal and thus, emphasizes a certain characteristic. The filter size represents the neuron's perception field. Mathematically, the filter's application to an input signal

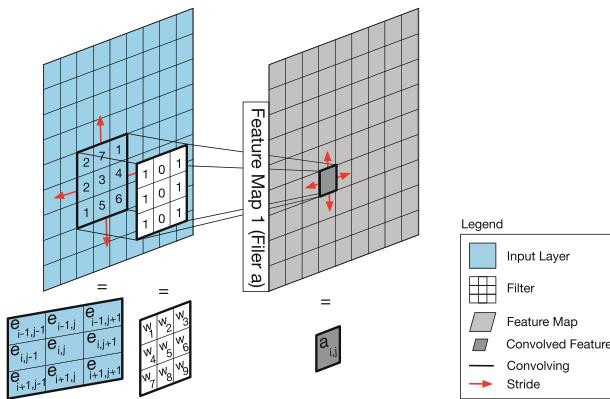


Figure 2.66 Convolutional Layer Architecture

is modeled by a discrete convolution from which the name CNN originates. Neurons reacting on the same kind of filter are grouped to a layer which is referred to as a *feature map* or the *convolutional layer* of a certain filter. Hence, the inclusion of different kinds of filters is a design activity which is vital for learning success.

Algorithmically, the filter runs over the entire feature space according to a *stride* and creates one feature map per filter. An example filter kernel having a size of 3x3 and providing the elements $w_1, w_2, w_3, \dots, w_9$ is visualized in Fig. 2.66. The red arrows visualize the kernel movement according to the stride defined.

Here, one can see how a filter is applied to an input signal $e_{i,j}$ and results in the CNN layer input $a_{i,j}$. For instance, input signals of an image are represented by input layer neurons visualized in blue. These are emphasized by a vertical 3x3 line filter and thus, result as the input signal for one neuron of the feature map visualized in gray. So, the CNN layer input $a_{i,j}$ of a feature map is derived and ready for activation as follows (Schwaiger and Steinwendner, 2019, p. 190):

$$\begin{aligned}
 a_{i,j} = & w_1 e_{i-1,j-1} + w_2 e_{i-1,j} + w_3 e_{i-1,j+1} \\
 & + w_4 e_{i,j-1} + w_5 e_{i,j} + w_6 e_{i,j+1} \\
 & + w_7 e_{i+1,j-1} + w_8 e_{i+1,j} + w_9 e_{i+1,j+1}.
 \end{aligned} \tag{2.109}$$

Activation Layers: As was defined in section 2.5.5.1, $a_{i,j}$ is activated by an activation function which is commonly the ReLU of Eq. 2.28 (Deru and Ndiaye, 2019,

p. 81). Since these are part of the ordinary working with ANNs, some visualizations consider the activation as part of the convolution layers.

Pooling Layers: The pooling layer holds results of the act of pooling and so, realizes the objective in reducing the dimensionality of the current layer's output, probably a convolutional layer. The principle of *pooling* is inspired by the neurobiological phenomenon of *lateral inhibition*, which means that active nerve cells inhibit the activity of neighboring nerve cells (Schwaiger and Steinwendner, 2019, p. 192).

Fig. 2.67 visualizes its algorithmic realization with the aid of a 3x3 filter and two successive pooling procedures. Features of an input layer are processed by the filter center according to a predefined *step size*, that is referred to as *stride*. Dependent on the *filter size*, neighboring features are selected and the green, orange, and red highlights in Fig. 2.67 represent the corresponding selections of three pooling steps. Out of the set of selected features, only one feature is overtaken as a sub-sample for the next layer. Within the next layer, the same color highlights the pooling result. As other features are not considered any further, the current activation dimensionality is minimized. This reduction can follow various kinds of pooling functions, such as the maximal value (*max pooling*) and the average value (*average pooling*), among others (Deru and Ndiaye, 2019, p. 82).

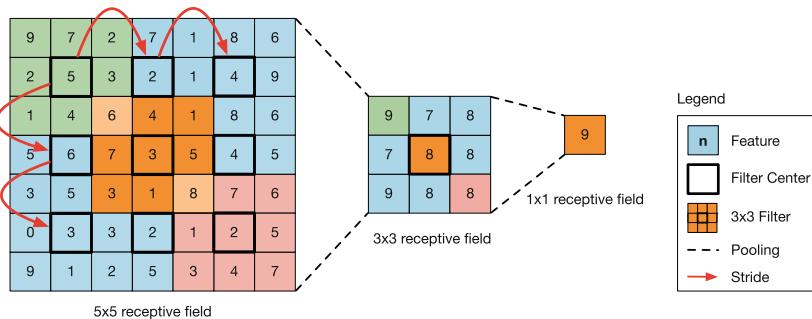


Figure 2.67 Pooling Layer Architecture (with a 3x3 Max Pooling Filter)

Within the figure presented, a max pooling is visualized which only leads to the takeover of the maximal value. As the act of pooling results in a reduced set of

data, which serve as input samples for consecutive layers, it is also referred to as *subsampling* or *downsampling* (Deru and Ndiaye, 2019, p. 82).

As a filter kernel reaches the edge of a feature space and demands for feature values, which are not provided within the original data, different *padding strategies* can be applied in order to generate missing values (Schwaiger and Steinwendner, 2019, p. 193). Here, one can find the *constant padding* providing a constant value, the *symmetric padding* mirroring the neighboring values with the aid of a mirror axis that sticks at the edge, or the *reflect padding* mirroring the neighboring values with the aid of a mirror axis that sticks to the last value.

Identification Layers: Commonly, CNNs provide convolutional layers and pooling layers, repetitively. Having reached the last layer of the detection part, neurons of the last kind of layers, and probably all pooling layers, are flattened so that they can serve as one dimensional input vector for ANN, as defined in section 2.5.5. Hence, the training of CNNs follows the same training procedures.

Critical Appraisal: Compared to ANNs described so far, CNNs consider spatial relations among neurons because of the modeling of receptive fields and the phenomenon of lateral inhibition with the aid of kernels being part of the convolving and pooling. MLPs, RNNs, and LSTMs do not consider spatial concepts at all.

2.5.8.5 Topological Networks

Aiming to consider the spatial arrangement of neurons, topological networks refer to a kind of ANN, in which the behavior of neurons is dependent on its neighboring neurons. Here, that region of a topological maps sticks out the most and is correlated best with a certain signal characteristic. This models the biological insight that many regions of the brain establish a linear or planar topology. Examples can be found in the sensory stimulus, which is positioned on the skin and at the retina, among others (Ritter, Martinetz, and Schulten, 1991, p. 68).

In contrast to an error-based ANN training (section 2.5.6), topological networks are trained on base of the *competitive learning* (section 2.5.6.2). Hence, they do not require a target classification and they realize an unsupervised learning (section 2.5.2).

Principally, topological networks can be found in Kohonen maps or self-organizing maps and neuronal gas. Each is presented in the following paragraphs.

2.5.8.5.1 Kohonen Feature Maps and Self-Organizing Maps

Self-organizing maps (SOMs) are also referred to as *Kohonen maps*, named after their inventor (Kohonen, 1989). Here, a *hard competitive learning* is carried out (c.f. section 2.5.6.2, Eq. 2.49–2.50) so that one winning neuron is identified and weights of this neuron and its neighboring neurons are adjusted.

SOMs provide the following architecture: every input neuron fully connected to neurons of the *Kohonen layer*; further, neurons of the Kohonen layer are fully meshed, but feedback cycles are not available. Fig. 2.68 visualizes the Kohonen network architecture.

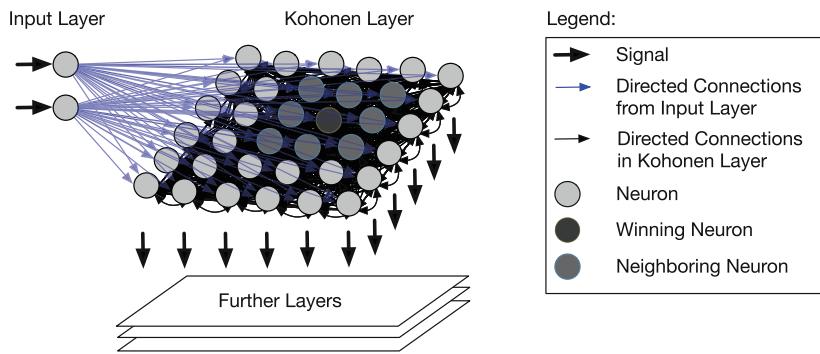


Figure 2.68 Kohonen Network Architecture

Here, one can see the spatial arrangement of the winning neuron and its neighboring neurons. Since neurons that have the same meaning are reinforced during the course of learning, neurons of the Kohonen layer tend to move together and regions of neurons evolve so that one can consider and refer to it as a map.

2.5.8.5.2 Neuronal Gas

In contrast to *Kohonen maps* and *SOMs* considered in section 2.5.8.5.1, a neuronal structure has evolved that is similar in its architectural aspects (Lämmel and Cleve, 2012, p. 253 ff., 264 ff.). It was introduced by Thomas Martinet and Klaus Schulten (Martinetz and Schulten, 1991). Here, a *soft competitive learning* is carried out (c.f. section 2.5.6.2, Eq. 2.51–2.54) so that one winning neuron is identified and weights of any neuron are adjusted depending on their distance to the winning neuron.

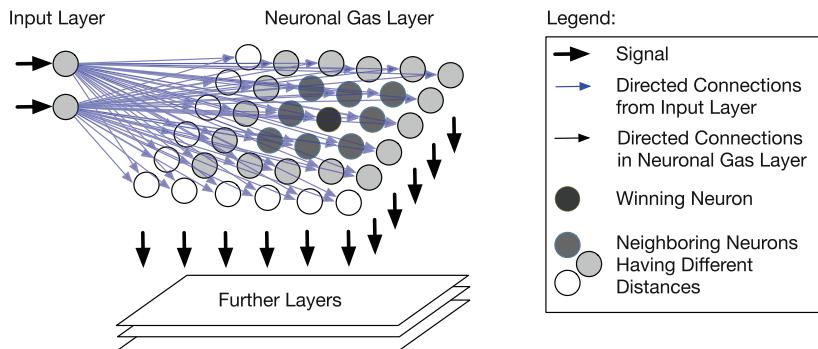


Figure 2.69 Neuronal Gas Architecture

Neuronal gas structures provide the following architecture: every input neuron fully connected to neurons of the *neuronal gas layer*; further, neurons of the neuronal gas layer are not meshed anyhow. Thus, the analogy of gas can be established here as Neurons of the input layer are only modeled in order to provide an input signal to the neuronal gas layer and neurons located here do not interact (Lämmel and Cleve, 2012, p. 264 ff.). Fig. 2.69 visualizes the neuronal gas architecture.

Here, one can see the spatial arrangement of the winning neuron and its neighboring neurons. Since neurons that have the same meaning are reinforced during the course of learning, neurons of the neuronal gas layer tend to move together. Thus, these act as learning topologies (Martinetz and Schulten, 1991; Martinetz and Schulten, 1994), and they are suited to time-series prediction (Martinetz, Berkovich, and Schulten, 1993), speech recognition (Curatelli and Mayora-Ibarra, 2000), image processing (Angelopoulou et al., 2005), and pattern recognition.

Variants of the simple neuronal gas architecture refer to *growing neural gas* (Fritzke, 1995), the *growing when required network* (Marsland, Shapiro, and Nehmzow, 2002), and also the *incremental growing neural gas* (Prudent and Ennaji, 2005).

2.5.8.6 Auto-associative Networks

Often, *auto-associative networks* might not be seen as ANNs since their structure and working behavior resembles the reaction of particles (particularly of heated metal particles). These cool down and, therefore, tend to lower the energy level of the metal substance until a stable state or rather a minimum is reached (Lämmel and Cleve, 2012, n 279). This kind of particular behavior was first transferred by

Hopfield to neuronal networks. Hence, Hopfield networks are first presented and then, as an evolution of Hopfield networks, Boltzmann machines are presented.

While being activated, these kinds of networks tend to converge to states that have been stored by the network, even if it is given only part of the input. Hence, the networks are suitable to recover distorted input and, further, they are suited to identify trained states that are similar to an input. Since the networks remember because of the similarity, they are referred to as associative memory (Ritter, Martinetz, and Schulten, 1991, p.47).

2.5.8.6.1 Hopfield Networks

Although Hopfield networks might not be seen as neuronal networks, they consists of neurons that are connected with each other. Further, they are activated by an input signal and produce an output signal with the aid of weights analog to ANNs. Hence, they can be considered as a network of neurons having a certain energy level of either one or zero.

As Fig. 2.70 shows, they follow a characteristic architecture. Without any layers, Hopfield networks consist of neurons that are fully meshed, but do not contain feedback cycles. Since Hopfield networks do not obtain any input and output neurons at all, input signals directly are fed into the network and the number of neurons within the network equals the size of the input signal.

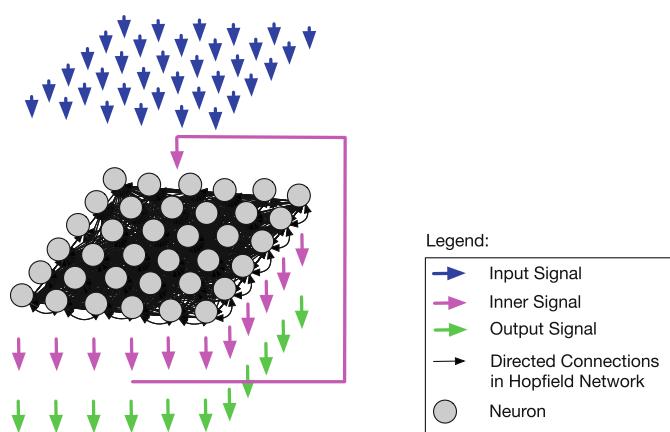


Figure 2.70 Hopfield Network Architecture

Forward Pass: When the network is initially activated by the input signal, one forward pass demands for a repetitive activation of the following: the network is repeatedly activated by its current activation up to the moment the activity of any individual neuron does not change anymore; then, a stable state is reached and the network's current activation is considered as network output.

Similarly to a forward pass of recurrent networks (Eq. 2.40), the neuron input a_h^t of a Hopfield network neuron h refers to the sum of the products of weights $w_{i,h}$ to the neuron h , and the input element x_i^t , at least at the first activation at time step t . Here, input from its neighboring neurons $b_{h'}^{t-1}$ refers to zero since the Hopfield network has not been activated before. The $w_{i,h}$ refer to one because no input neurons are available at all. So, input is directly injected in the Hopfield network neurons.

Then, for consecutive time steps, no further input x_i^t is provided so that the neuron input a_h^t of a Hopfield network neuron h refers to the sum of the products of weights $w_{h'h}$ between the neurons h and h' and the output $b_{h'}^{t-1}$ of Hopfield network neurons h' of previous time steps $t-1$. The corresponding activation b_h^t of a neuron h of a Hopfield network, therefore, follows the application of an activation function $\theta_h(\dots)$ analog to Eq. 2.35 (Lämmel and Cleve, 2012, p. 281):

$$b_h^t = \theta_h(a_h^t) = \begin{cases} 1, & \text{if } a_h^t > \theta_h \\ 0, & \text{if } a_h^t < \theta_h \\ b_h^{t-1} & \text{if } a_h^t = \theta_h. \end{cases} \quad (2.110)$$

Depending on the threshold θ_h of a neuron h , the neuron is either firing or it isn't. Only when the change of a neuron's state does not contribute to stabilize, for instance, because it already is stable, the neuron is not affected.

Energy-based Interpretation: Since a neuron fires with regard to the threshold θ_h , the energy level of the Hopfield network is reduced with every network activation so that at least a local minima is achieved when the network is stable (Lämmel and Cleve, 2012, p. 287). This can be measured by a problem-specific energy function:

$$E^t = -\frac{1}{2} \cdot \sum_h \sum_{h'} w_{h,h'} \cdot b_h^{t-1} \cdot b_{h'}^{t-1} + \sum_h \theta_h \cdot b_h^{t-1}. \quad (2.111)$$

The repetitive activation procedure varies in the way network activations are generated (Lämmel and Cleve, 2012, p. 281, 282). Either all activations are generated before the next activation iteration is reached, known as *synchronous proceeding*, or

activations are generated randomly, without respecting the number of times a certain neuron has been activated before, known as *asynchronous proceeding*. Here, it is essential to distinguish time which results because of the iteration of the activation, and time which results because of the repetitive activation within one iteration, since the synchronous proceeding asks for various activations during one time step. Here, when aiming to consider Hopfield networks in the course of process simulations, one needs to establish an interpretation.

Training: In contrast to ANN training approaches, weights of Hopfield networks can easily be trained by design. One can say that the learning including the network's weight adjustment is carried out by a one-shot design activity which reflects the learning problem (Ritter, Martinetz, and Schulten, 1991, p. 49). If p represents the number of patterns m the Hopfield network has to recognize, the weight $w_{h,h'}$ between the Hopfield network neurons i and j results with

$$w_{h,h'} = \begin{cases} \sum_p m_{p,h} \cdot m_{p,h'}, & h \neq h' \\ 0, & h = h' \end{cases}. \quad (2.112)$$

With this, the capacity of the network is delimited by the number of neurons within the network and one needs about seven neurons to store one pattern (Lämmel and Cleve, 2012, p. 285). Training variants refer to adaptations of the Hebbian learning rule applied to Hopfield networks (McEliece et al., 1987), the pseudo-inverse rule not being local and incremental (Kanter and Sompolinsky, 1987), the Storkey learning rule considering a local field at neurons (Storkey, 1997), or a Backpropagation version (Ritter, Martinetz, and Schulten, 1991, p. 53).

2.5.8.6.2 Boltzmann Machines and Simulated Annealing Machines

Boltzmann machines were first designed by Geoffrey Hinton and Terrence J. Sejnowski in 1985 (Ackley, Hinton, and Sejnowski, 1985). Providing the same architectural structure and applying the same technical realization as Hopfield networks, Boltzmann machines additionally consider a possibility to exit local energy minima so that the chance to achieve a global optima is raised.

Forward Pass: Following the metaphor of the particle behavior of metal, this resembles the material temperature T under which the metal is processed: the higher its temperature, the easier its forming will be. Therefore, for Boltzmann machines, the activation function refers to

$$b_h^t = \theta_h(a_h^t) = \begin{cases} 1, & \text{if } \text{random}() \leq \frac{1}{1+e^{-\frac{a_h^t}{T}}} \\ 0, & \text{else} \end{cases}. \quad (2.113)$$

Here, the behavior to fire or not to fire is changed by a probabilistic component. So, the neuron is made to fire although it does not want to fire or it is inhibited for firing although it wants to fire, both of which results in a systematic worsening of the current energy level (Eq. 2.111) and is accepted because of the intention to identify the global optima.

Kinds of Machines: If T remains constant, refer to *Boltzmann machines*, and if T is not constant, one refers to *simulated annealing machines* because of the analogy of repetitive warming ups and cooling downs during steel forging (Lämmel and Cleve, 2012, p.293). Typically, the activation of these kinds of neuronal networks starts with high T values, since the probability to make neurons firing or inhibit them for the firing is equal. Later activation runs t reduce the current activation function by

$$T = \frac{1}{t}. \quad (2.114)$$

when the average of the energy level is constant during asynchronous activations (see section 2.5.8.6.1). This reduces the probability to make the neuron behave irregularly. So, the temperature is lowered and the energy level can be reduced by repetitive activations in more suitable energy regions. Hence, Boltzmann and simulated annealing machines can be considered as a network of neurons having a certain energy level of either one or zero and behaving in a stochastic manner.

Here, it is essential to distinguish time in three ways: first, time resulting because of the iteration of the activation; second, time resulting because of the repetitive activation within one iteration, since the synchronous proceeding asks for various activations during one time step; and third, time resulting because of the decrease of the temperature. Here, if one aims to consider Boltzmann machines and simulated annealing machines in the course of process simulations, one needs to establish an interpretation.

Variants: Variants of Boltzmann machines refer to restricted Boltzmann machines (RBMs) following the same structure as Boltzmann machines, but differentiating in visible and invisible neurons and removing inner connections among the set of visible neurons and the set of invisible neurons (Salakhutdinov, Mnih, and Hinton, 2007; Hinton, 2012). Further, one can find deep Boltzmann machines (DBMs)

having multiple layers of hidden random variables satisfying Markov properties (Hinton and Salakhutdinov, 2012).

2.5.9 ANN Modeling

In accordance with the model understanding established in section 2.1.1.1 with Def. 1 and the modeling understanding established in section 2.1.1.4 with Def. 5, the following focuses on the construction of neuronal models (Def. 24) with the aid of a modeling language. Analogous to Def. 8, the ANN modeling language (ANN ML) is defined to be:

Definition 29 (ANN Modeling Language).

An artificial neuronal network modeling language (synonymous with ANN modeling notation) is a not necessarily, but mostly, graphical notation satisfying more or less sophisticated syntactic rules to be applied by a model creator to support the construction of a model creator's reality and create a representation of its subjective or objective reality about neuronal networks, the so-called ANN model, within the creator-specific subjective or objective model environment on arbitrary levels of abstractions and granularities of deep structures.

Beside the structural description of ANN, this includes dynamic effects such as biological mechanisms and their behavior during the course of activation, such as by simulation scenarios, as well as knowledge use, generation, transfer, and diffusion. By this, the ANN ML must stand for modeling of biological neuronal networks (neuroscience) as well as artificial neuronal networks (machine learning science).

Principally, ANN modeling can be carried out by the following ways. First, ANN models can be constructed manually by creating code. This includes code from programming libraries (section 2.6.2.2–2.6.2.5) as well as code abstractions for its visualization issued here. Second, ANN models can be constructed by algorithms with the aid of tools that provide a graphical user interface and modifiable parameters controlling the construction. These kinds of tools are issued in section 2.6.1.8 as well as section 2.6.2.6–2.6.2.9. Third, ANN models can be constructed by ANN modeling languages that represent graphical abstractions of ANN models and have the potential for an interactive model construction by click and drag options. This most resembles the principles of a process modeling.

As OoI evaluations will show, the first two options are commonly exhausted and a potential can be identified at the third option. As the first two options are issued in section 2.6, the following puts focus on graphical modeling languages.

Selection of representatives: As a great number of ANN modeling notations can be found in the literature, the selection of adequate works must be addressed. Attempts identifying and designing best candidates include Gleeson et al. (2010), Cannon et al. (2014), Abadi et al. (2016), and Wongsuphasawat et al. (2018). The contribution, therefore, selects ANN MLs according to the following criteria:

1. Modeling languages having a broad market distribution which was determined by the SLR conducted.
2. Modeling languages having a broad acceptance in the modeling community.
3. Modeling languages being applied in at least one scientific publication.

Adequate examination level: In order to satisfy requirements for an adequate dealing with objects of investigation (OoI) in terms of SLR (Levy and Ellis, 2006), which corresponds to the application of Bloom's taxonomy (Bloom et al., 1956), each modeling language is described by the following categories:

- The short *abstraction*, which serves as a kind of introduction to modeling languages on the same abstract level.
- Its individual *description*, which characterizes the individual OoI.
- *Side facts* issuing the OoI-specific development and historic evolution.
- The provision of a *meta-model* which visualizes modeling items and its underlying syntax. This is completed by *modeling conventions* clarifying how the meta-model of the OoI is applied so that concrete process models are constructed.
- An *example* visualizes the appliance and underlines modeling strengths of the OoI.
- Finally, the *critical appraisal* evaluates if and how the individual modeling language is suited to stand as the foundation for the CoNM.

So, according to Bloom's taxonomy, categories considered for any modeling language correspond to the levels of *knowing* and *comprehending* if OoIs are described and side facts are presented, *applying* as a meta-model is generated and an example is presented, *synthesizing* as a common level of understanding is worked out, as well as *analyzing* and *evaluating* as common analysis criteria are issued and the process modeling languages are regarded with respect to the CoNM.

2.5.9.1 TensorFlow Graphs

Besides the programming library called TensorFlow (section 2.6.2.5), a corresponding modeling language is provided called *TensorFlow graphs*. It intends to generate

graph representations of TensorFlow code models visualizing the ANN structure as well as information about low-level operations (Wongsuphasawat et al., 2018). As it provides a powerful set of modeling principles and modeling objects, TensorFlow graphs are considered as an individual OoI.

Description: TensorFlow's graphs represent computation, shared state, and operations mutating states (Abadi et al., 2016, p.265). As these further map the nodes across many machines, the graphs issue the distributed processing in a computer cluster within a machine including multicore CPUs and GPUs and custom-designed ASICs.

Each node or vertex represents a unit of local computation or what is called *tensor* (Abadi et al., 2016, p.270). Here, one can find four different kinds of representations: first, representations for *mathematical functions* (Deru and Ndiaye, 2019, p.148); second, representations for *constant, sequence, and random operations* for initializing tensor values; third, representations for *summary operations* for producing log events for debugging; and fourth, representations for *variable operations* for storing model parameters (Wongsuphasawat et al., 2018, p.3).

Each edge or connection represents one of the following three (Wongsuphasawat et al., 2018, p.3). First, *data* being modeled as tensor or more precisely, n-dimensional arrays of elements with primitive type, such as *string*, *int32*, or *float32* (Abadi et al., 2016, p.3). These serve as the input and output of operations and are visualized by a solid arrow. Second, *references* representing pointers to variables rather than values. Here, a yellow arrow serves as the modeling element. Third, relations issuing the *control dependency* (dotted arrows), so that a control flow can be established (Abadi et al., 2015, p.8; as well as Abadi et al., 2016, p.271). Examples here refer to conditions for the start of an operation.

Further, on top of the structures defined so far, beside an API-based denomination, elements can obtain an individual name and graphs can be clustered by individual namespaces (Wongsuphasawat et al., 2018, p.3). By doing so, a hierarchy can be established that allows the expansion and reduction of namespaces. So, elements of more detailed namespaces are hidden and the visualization of transparent models is supported.

The transparency of the model is further increased by the following three reorganizations (Wongsuphasawat et al., 2018, p.4). First, non-critical operations are extracted. These can refer to summary and constant operations. Second, a clustered graph with bundled edges is built. Besides building the hierarchy of namespaces already mentioned, this includes the bundling of edges so that drawing edges between all visible notes can be avoided. Third, auxiliary nodes are extracted from

the clustered graph. These are replaced by small proxy icons within the model and positioned on the right of the layout.

On behalf of modeling elements described so far, the *overview* of the model is constructed. Further, on behalf of color-based overlays, various kinds of views are derived (Wongsuphasawat et al., 2018, p. 7). First, the *structure view*, in which group of nodes having an identical structure are highlighted by the same color. Second, the *device distribution view*, in which each operation is colored according to the device it is allocated to run on. Third, the *compute time view*. Here, nodes are colored in dependency of the time required for computation. Fourth, the *memory view* showing nodes with high degree of saturation if high memory usage is required for computation. By the latter two kind of views, computational bottlenecks shall become apparent.

Side facts: Being provided by the TensorFlow repository (Google-Brain-Team and Community, 2018b), TensorFlow graphs are licensed under the *Apache 2.0* Open Source software license. Originally, it was developed for the internal use of Google's AI laboratory and was developed by its community (Abadi et al., 2015; Abadi et al., 2016; Wongsuphasawat et al., 2018).

Meta-model: Fig. 2.71 provides the general *TensorFlow Graph meta-model*. Here, it becomes clear, how incoming (on the left) and outgoing (on the right) modeling objects are related in the TensorFlow graphs.

Considering the presented general ANN model of the TensorFlow graphs, ANNs are constructed by the following TensorFlow graph design rules (derived from Abadi et al., 2015; Abadi et al., 2016; Wongsuphasawat et al., 2018):

1. TensorFlow Graph's core node refers to *operations* and *mathematical functions*, the so-called *tensors*.
2. The namespace's denomination should follow a meaningful characterization. Inner structures can be unveiled by expansion or hidden by reduction.
3. Edges connect two tensors corresponding to their sequence of action so that they are directed and the *data flow* (gray solid line), the *control dependency* (gray dashed line), or the *reference* relation (orange line) becomes transparent.

As the TensorFlow Graph standard is strictly formal, the guidelines presented here must be seen as interpretation for an algorithmic realization.

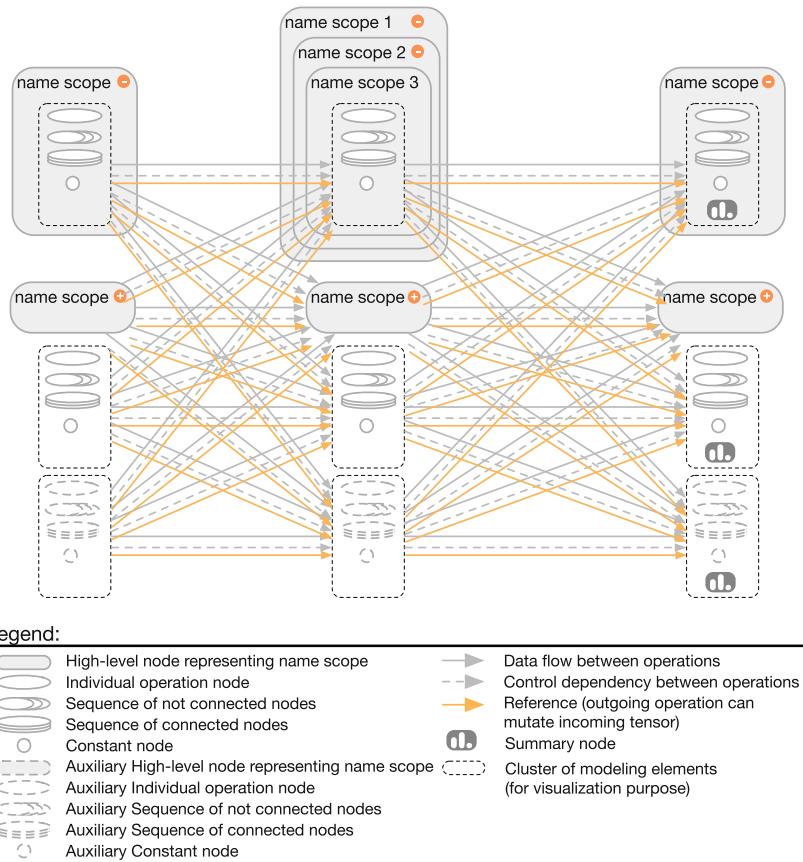


Figure 2.71 Generic TensorFlow Graph Meta-Model (synthesized from literature)

Example: A TensorFlow Graph example can be found in Fig. 2.72. It shows a simple four-layer sequential classification model example visualized as TensorFlow Graph with the tool called TensorBoard (Google-Brain-Team, 2019).

Within the visualization tool, one can find the structure view of the TensorFlow graph. Since, by structure, similar nodes show the same color, one can find redundant structures at nodes *flatten* and *flatten_1* as well as *dropout* and *dropout_1*. The latter two nodes have been expanded by interactive double clicking so that its detailed

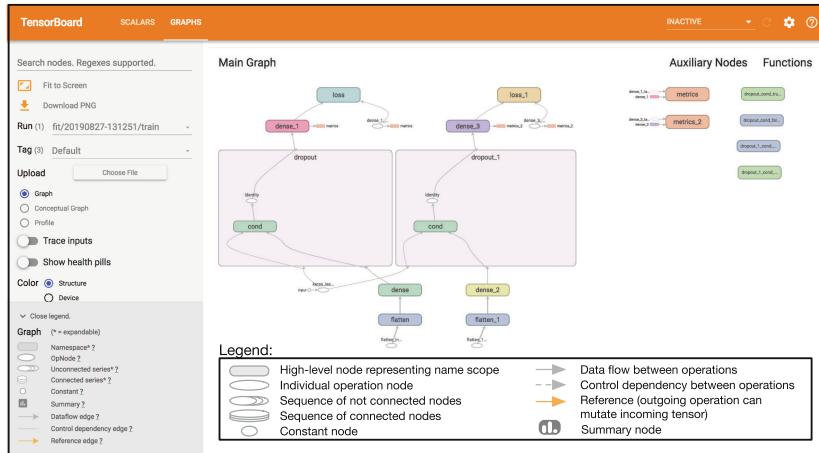


Figure 2.72 TensorFlow Graph Example by TensorBoard (created to show the strength of the object of investigation)

yet same structural elements are unveiled. Further, auxiliary nodes *metrics* and *metrics_2* have been extracted via the context menu opened using a right click so that these can now be found on the right of the model.

Critical appraisal: In general, TensorFlow graphs focus on the dataflow visualization of TensorFlow-based code models. As the ANN context is established successfully, information flows can be considered adequately. Regrettably, different forms of knowledge are not considered. Clearly, machine learning issues are addressed, such as the debugging of ANN structures such as their performance evaluation during the course of training and testing, and computer capacities, among others. However, modeling language criteria from the knowledge management domain are not addressed adequately.

A data set specification, as it is required for ANN learning or fed by simulators, is not enabled either. This includes data generated by ANNs during the course of simulation. TensorFlow graphs are further limited to the visualization of the inner dataflow. Their connection to real-world data streams by design is not supported at all. By this, modeling language criteria from the simulation domain are not met either.

Furthermore, TensorFlow Graph's focus lies on the visualization of ANN structures and these are not considered to be part of simulations or part of business contexts' modeling language criteria, aside from the ANN domain, such as from business process context, organization, behavior, and function perspectives. Additionally, the simulation tool criteria cannot be met. Even modeling language elements from the ANN domain can be only considered if these are translated into TensorFlow-based operations. In contrast to other ANN modeling languages, such as NeuroML (section 2.5.9.2) or LEMS (section 2.5.9.3), TensorFlow graphs do not provide a collection of biological mechanisms. Hence, NN criteria are not met adequately.

When dealing with TensorFlow graphs, interaction is limited to the expansion or reduction of namespaces and the disregard of selected graphical elements. This does not correspond to an interactive model construction and neglects a cooperative modeling in the sense of process modeling. Hence, important modeling tool criteria are not satisfied.

Although modeling elements of TensorFlow graphs do not principally exclude an ANN design by dragging and dropping modeling elements within the graphical modeling space, its visualization tools, such as TensorBoard, do not allow a bidirectional information exchange between the code model and graphical model. Before a graphical representation can be constructed, first, the code is set, and then, after the TensorFlow function *tf.summary.FileWriter* has created a text-based graph, its visualization can be interpreted and displayed (Schwaiger and Steinwendner, 2019, p. 211). Hence, important ANN tool criteria are not satisfied. This includes a 3D positioning, the use of AR techniques, and spatial positioning.

Although the modeling tool TensorFlow Graph provides a clear focus on machine learning perspectives and essential biological mechanisms have not been addressed adequately, the modeling concept is suitable to be considered for the CoNM construction since it addresses essential subjects of ANN modeling. This regards the processing across systems, its KPI overlays over the basic model, and the recognition of similar structures, as well as its dealing with namespaces. For the realization of tasks across systems, further global optimization objectives over scenarios still must be addressed (Grum et al., 2018). In combination with ANN modeling approaches and efficient task distribution approaches, considering simulation perspectives and biological mechanisms, TensorFlow can stand as a suitable foundation for the CoNM construction.

2.5.9.2 NeuroML

NeuroML is an neuronal modeling language that enables the data-driven model description of neurons and neuronal networks on a high level of detail (Gleeson et al.,

2010). While neuronal simulators such as GENESIS (section 2.6.2.9), NEURON (section 2.6.2.8), NeuroConstruct (section 2.6.2.6), NEST (Diesmann and Gewaltig, 2002; Brette et al., 2007), MOOSE (Ray and Bhalla, 2008), or PSICS enable the easy specification of neuronal models that are not interoperable, NeuroML intends to specify neuronal models and compartments as a standalone form so that their reuse, accessibility, and cross-simulator validation are enabled.

Description: As the modeling granularity of neuronal systems varies greatly in detail (Herz et al., 2006), a three level structure of NeuroML separates model descriptions into the anatomical structure and various biological mechanisms as they are supported by neuronal simulators (Gleeson et al., 2010) and have been described in section 2.5.1.

The first level builds on *MorphML* (Crook et al., 2007) and allows the description of cell structures ranging from single compartment cells, multicompartment cells, as well as detailed cells based on morphological reconstructions.

The second level builds on *ChannelML* and allows the use of level 1 descriptions. Here, passive properties are described side by side with location and density properties of active conductances on the cell. Further, membrane processes are described that generate the electrophysiological behavior of the cells.

The third level builds on *NetworkML* and allows the description of networks of these neuronal models. This includes the 3D placement of cells and their synaptic connections.

As these levels can be used in isolation and jointly in order to describe relevant details, models of voltage- and ligand-gated conductances, as well as models of electrical coupling, synaptic transmission, and short-term plasticity, can be established (Gleeson et al., 2010). This includes the specification of morphologically-detailed models from individual neurons as well as highly detailed cortical network models.

Side facts: Building on Gardner et al. (2001), the concept for NeuroML was originally developed by Goddard et al. (2001) and has been published as an Open Source project (Davison et al., 2005). Following successful standardization initiatives, such as system biology markup language (SBML) (Hucka et al., 2003), CellML (Lloyd, Halstead, and Nielsen, 2004a), BrainML (Gardner et al., 2008), and MathML (ISO Central Secretary, 2016), NeuroML specifications are based on XML (Bray, Paoli, and Sperberg-McQueen, 1997) which has been identified as a language used for successfully exchanging structured information between computer applications (Gleeson et al., 2010).

Model Definition & Model Instantiation

```
<element_a attribute_a_1...m="X" optional_attribute_a_1...n="X">
  <element_b attribute_b_1...o="X" optional_attribute_b_1...p="X">
    <element_c attribute_c_1...q="X" optional_attribute_c_1...r="X">
    ...
    <element_d attribute_d_1...s="X" optional_attribute_d_1...t="X">
</element_a>
```

Legend:

abc - element denomination	abc - parameter
abc - attribute denomination	abc - optional attribute denomination

Figure 2.73 Generic NeuroML Meta-Model (synthesized from literature)

Meta-model: Fig. 2.73 provides the general *NeuroML meta-model*. Here, it becomes clear how a hierarchical structure of modeling objects is built in the NeuroML standard.

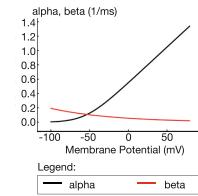
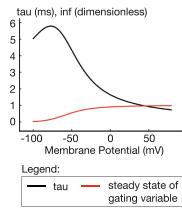
Considering the general ANN model of the NeuroML presented here, ANNs are constructed by the following NeuroML design rules (Cannon et al., 2014; Gleeson et al., 2010):

1. NeuroML's core entries refer to *elements*.
2. Elements are characterized by an arbitrary high number of *attributes*, *optional attributes* or by *nested elements*.
3. NeuroML-based ANN models are stored in separate XML files.

As the NeuroML standard is strictly formal, the guidelines presented here must be seen as an interpretation for an algorithmic realization.

Example: A NeuroML example description is visualized in Fig. 2.74. It shows the Hodgkin-Huxley formalisms for K⁺ conductance model with four channels (Hodgkin and Huxley, 1952) that has been described by NeuroML conventions (Gleeson et al., 2010 under CC-BY 4.0). Since each instance demands for identical gating mechanisms with open and closed states, the transitions are described on behalf of the ChannelML listing.

On the right, one can find the NeuroML listing using ChannelML standard, which specifies two kinds of transitions: first, the alpha transition representing forward transition rates from n0 to n (line #11 – #13), and second, the beta transition representing reverse transition rates from n to n0 (line #14 – #16). The corresponding visualization of both rates can be found in Fig. 2.74 (a). Using the listing presented,

(a) Transition Rates for K^+ Channels.

(b) NeuroML-based Simulation, e.g. with NeuroConstruct.

```

1 <channelml xmlns="http://morphml.org/channelml/
2   schema" units="Physiological Units">
3   <channel_type name="HH_KConductance">
4     <status value="stable"/>
5     <current_voltage_relation cond_law="ohmic"
6       ion="k" default_gmax="36"
7       default_erev="-77.0">
8       <gate name="n" instances="4">
9         <closed_state id="n0"/>
10        <open_state id="n"/>
11        <transition name="alpha" from="n0" to="n"
12          expr_form="exp_linear" rate="0.1"
13          scale="10" midpoint="-55"/>
14        <transition name="beta" from="n" to="n0"
15          expr_form="exponential" rate="0.125"
16          scale="-80" midpoint="-65"/>
17      </gate>
18    </current_voltage_relation>
19  </channel_type>
20 </channelml>
```

LISTING 2.1: NeuroML Description Example with ChannelML.

Figure 2.74 NeuroML Example for a Description of the Hodgkin-Huxley Type K^+ Conductance Model (created to show the strength of the object of investigation)

a neuronal simulation can be instructed, such as *NeuroConstruct* (section 2.6.2.6), so that the ANN structures are applied in simulation engines holding their own mechanism implementations. By this, the time constant τ_{au} of the transition or the steady state of a gating variable can be visualized, as Fig. 2.74 (b) shows.

Critical appraisal: Although NeuroML descriptions have been proven by the use of five independently developed simulators (Gleeson et al., 2010), and obviously support the cross-validation, model reuse, etc., the simulation modeling is limited to the structural description of ANN structures and the functional use of mechanisms.

The NeuroML strength particularly refers to the option to describe biological details on a high granularity and to subsequently access behavioral mechanisms of textual descriptions and mechanisms being available in neuronal simulators (Vella et al., 2014). The mechanism realization includes the graphical interpretation applying to the simulation tools.

Further, the specification of plots, simulation controlling, and data management are not issued at all. Thus, only basis simulation modeling language criteria are satisfied.

Going beyond a graphical interpretation of ANN structures, this was not intended by NeuroML as even a graphical representation of models being described by XML is missing. Therefore, a describing of XML code lines refers to a rather technical challenge (or on behalf of third party tools) than to an interactive modeling as it is known from the process modeling domain. Further, the cooperative modeling is not enabled at all so that modeling results can be discussed by the community as they are published in repositories and model stores. Therefore, common modeling and simulation tool criteria are not met anyhow. This includes the technical representation of data and current ANN structures: a database-based organization is not issued at all. A 3D visualization of an ANN with augmented reality (AR) techniques is not regarded either.

Modeling language criteria coming from the business process domain, where one can find business process context, organization, behavior and function perspectives, and modeling language criteria coming from knowledge management, where one can find information and knowledge perspectives, are not issued further.

Although aiming to define models from biology systems and mapping them to simulators from the neuroscience-domain, NeuroML majorly focuses on NN and ANN structures. Therefore, modeling language criteria from the ANN domain are satisfied and even a 3D positioning is considered by NeuroML.

For the CoNM, the optional description of various granularity levels is particularly ideal. Further, NeuroML issues key compartments of ANNs which must be considered by the CoNM construction. Additionally, when intending to serve various tools by the CoNM, the accessibility of the CoNM modeling concept need to provide interfaces to various tools. Here, the NeuroML can stand as an example. Although XML has been proven yet again with NeuroML to be an efficient data exchange standard, this does not correspond to a database-based structuring standard of the process domain and here, can be considered as an alternative.

2.5.9.3 LEMS

Low Entropy Model Specification (LEMS) language is an neuronal modeling language that enables the description of physio-chemical systems, such as neurons and neuronal networks, and intends to be flexible for the consideration of new domain-specific concepts, as well as to avoid redundancy on behalf of a machine readable representation. The name LEMS was derived from the idea of a *Low Entropy Model Specification*, which refers to the separation of the general definition of model components and the corresponding instances with particular parameter values (Cannon

et al., 2014). Thus, higher level domain-specific languages such as NeuroML (section 2.5.9.2) can be built as a second version.

Description: In contrast to *domain-specific description languages*, such as NeuroML as version 1 (Gleeson et al., 2010) or the SBML (Hucka et al., 2003), LEMS jointly considers data and logic that is required to fully describe a model. Thus, data as well as logic are not spread across the following three (Cannon et al., 2014): first, the model descriptions; second, the documentation about the modeling description language; and third, different simulation engines. In cooperation with NeuroML version 2, LEMS includes logic and functions as description level for mechanisms having been present at neuroscience-specific simulators to date.

In contrast to *generic model description languages* such as CellML (Lloyd, Halstead, and Nielsen, 2004b), which allow the unambiguous mathematical representation of the model without any further domain-specific knowledge, LEMS does not hamper the flexibility to apply models in different simulation engines. This is because models are not represented and constructed in different ways. Instead of being challenged by dealing with models from various sources, including the combination of models, the implementation with regard to the application's compliance, and their validation (Cannon et al., 2014), a common description base is established. This can simply be included by simulation engines.

As LEMS focuses on the specification of Reproducible, Accessible, Portable, and Transparent (RAPT) models that are used by higher level languages, a non-redundancy model description can be established. This is realized with a *ComponentType* expressing structure and dynamic behavior (Cannon et al., 2014, p. 11–12). Then, the instantiation is realized with the aid of higher level languages showing concrete parameters.

Side facts: In an effort to overcome shortcomings of NeuroML as version 1, LEMS was developed by Cannon et al. (2014). It has been published as an Open Source project (Cannon et al., 2018) under [GNU-GPL 3.0](#) and is oriented, but not limited, to specifications of XML (Bray, Paoli, and Sperberg-McQueen, 1997).

Meta-model: Fig. 2.75 provides the general *LEMS meta-model*. Here, it becomes clear how the hierarchical structure of modeling objects is defined by one file and the corresponding instantiation is build by another file in the LEMS standard.

Considering the general ANN model of the LEMS presented here, ANN descriptions are constructed by the following LEMS design rules (Cannon et al., 2014; Vella et al., 2014):

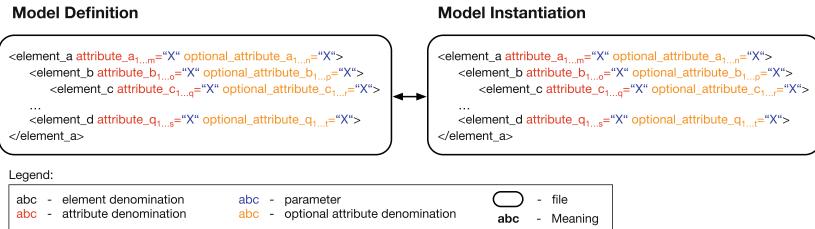


Figure 2.75 Generic LEMS Meta-Model (synthesized from literature)

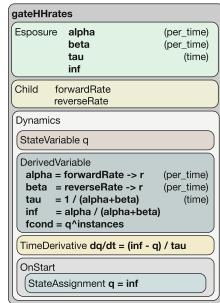
1. LEMS's core entries refer to *elements*.
2. Elements are characterized by an arbitrary high number of *attributes* with default parameters, *optional attributes* with default parameters, or by *nested elements*.
3. LEMS elements are instantiated by defined *attributes* with concrete parameters, *optional attributes* with concrete parameters, or by *nested elements*.
4. LEMS-based ANN models are stored in separate files.

As the LEMS standard is strictly formal, the guidelines presented here must be seen as interpretation for an algorithmic realization.

Example: A LEMS example description is visualized in Fig. 2.76. It shows the Hodgkin-Huxley formalisms for K⁺ conductance model with four channels (Hodgkin and Huxley, 1952) that has been described by LEMS conventions (Cannon et al., 2014; Bowen, Gleeson, and Larson, 2018; Bowen et al., 2019 under CC-BY 4.0).

On the left, in Fig. 2.76 (a), one can see the generic definition of voltage gated channels being specified by the following variables: first, the alpha transition representing forward transition rates; second, the beta transition representing reverse transition rates, the corresponding visualization of both rates can be found at Fig. 2.74 (a) showing the same biological example; third, the time constant *tau* of the transitions; and fourth, the steady state of gating variables. Beside these derived variables, the time derivative is described along with the state assignment of the simulation initialization (*onStart*).

On the right, in Listing 2.2, one can see the instantiation of four Potassium channels called *kChan* with the aid of a NeuroML description. Since basic mechanisms have been specified with LEMS, the alpha transitions can be instantiated (line #7 – #9) as well as the beta transitions (line #10 – #12), the channel types (line #4 – #5),



(a) LEMS Description.

```

1 <neuroml xsi:schemaLocation="http://www.neuroml.
2           org/schema/neuroml2 ..."
3           id="kChan">
4   <ionChannelHH id="kChan" conductance="10pS"
5     species="k">
6     <gateHHrates id="n" instances="4">
7       <forwardRate type="HHExpLinearRate"
8         rates="0.1per_ms" midpoint="-55mV"
9         scale="10mV"/>
10      <reverseRate type="HHExpRate"
11        rates="0.125per_ms" midpoint="-65mV"
12        scale="-80mV"/>
13    </gateHHrates>
14  </ionChannelHH>
15 </neuroml>

```

LISTING 2.2: LEMS-based Description Example with NeuroML vers. 2.

Figure 2.76 LEMS Example for a Description of the Hodgkin-Huxley Type K⁺ Conductance Model (created to show the strength of the object of investigation)

and the number of instances (line #6). A neuronal simulator can be instructed with this, such as *NeuroConstruct* (section 2.6.2.6) or PyLEMS (Vella et al., 2014), so that instantiated ANN structures are applied in its simulation engines. These can either operate on the basis of mechanisms described by LEMS or by its own mechanisms (Cannon et al., 2014, p. 12). By both kinds of simulation mechanism representations, the time constant *tau* of the transition or the steady state of a gating variable can be visualized, as Fig. 2.74 (b) shows.

Critical appraisal: LEMS strengths clearly lie in the description of dynamic effects having been provided by various tools and different representations mentioned insofar. Thus, a common understanding is supported since LEMS provides a common description base for mechanisms to be applied in simulation engines. With this, common ANN modeling language criteria are met required for the simulation. As LEMS does not intend to replace simulators, a corresponding simulation control and scenario design is not supported. Hence, simulation modeling language criteria are not met adequately. This includes simulation data generated during the course of simulation. Further, graphical interpretation such as a 3D or AR visualization is not part of LEMS. At the very least, the 3D positioning of ANN compartments can be described.

The technical dealing with LEMS-based descriptions still is challenging because of the strict formalization of LEMS syntax and failure-non-tolerant interpretation

of human input for technical systems. Although LEMS-specific interpreters, such as libNeuroML, PyLEMS (Vella et al., 2014), or jLEMS (Cannon et al., 2014, p. 7), enable the dealing with descriptions without needing third party simulators, LEMS-based descriptions are typically built with third party tools (Cannon et al., 2014, p. 13) providing visualizations. In LEMS, a graphical and interactive describing or modeling, as it is known from process modeling tools, is not enabled. Further, a cooperative modeling by several experts is not supported. So, a modeling rather refers to an isolated description and results might be discussed via communities. Hence, modeling tool criteria are not satisfied.

Further, no perspective from the business process domain or knowledge management domain are considered in LEMS. Common modeling language criteria, therefore, are not met at all by LEMS.

As the CoNM does not intend to cross-validate concepts by various simulators, the modeling of dynamics is not necessarily a requirement for the CoNM construction. Rather, it rather can be seen as a kind of empowerment since it gains access to different simulation perspectives and third party simulators. Thus, it so can increase the CoNM diffusion within the research community.

2.5.9.4 Critical Appraisal of ANN MLs

Faced with the aforementioned ANN modeling languages and its interim conclusions, it seems that today's ANN model description approaches are able to consider numerous aspects from highly specialized contexts such that particularly certain aspects are considered optimally. While TensorFlow graphs enable the modeling of ANN structures, their operations, and data inclusion from an ANN perspective, the modeling on various levels of granularity is lacking. This worsens the biological plausibility and puts a focus on techniques enabling the dealing with ANN. NeuroML and LEMS enable the realistic modeling of ANN structures at different levels of granularity from a biological perspective. Further, LEMS focuses on behavioral and dynamic mechanisms more than NeuroML. As this is realized by the inclusion of mechanisms within the model, final models are more complex. In comparison with TensorFlow graphs, this does not consider data organization aspects adequately.

Individually, they are still immature and do not consider sophisticated ANN techniques. Further, they do not consider the following process modeling aspects:

- None of presented ANN ML approaches considers neuronal modeling objects on an all-encompassing basis. This includes ANN structures, dynamic behavior, and biological mechanisms as well as computational realization and computing KPIs. Therefore, an integration of ANN objects with currently available ANN concepts is missing.

- None of presented ANN ML approaches considers knowledge management objects. Therefore, an integration of ANN objects with currently available knowledge transfer concepts is missing. This includes the generation, distribution, diffusion, and transfer as well as forgetting of knowledge. Further, an integration of ANN ML objects with currently available psychological learning concepts is missing.
- None of presented ANN ML approaches considers actual entities. Therefore, an integration of ANN mechanisms with currently available process modeling approaches is missing. This includes common processes, knowledge management processes, and organizational aspects. Further, an integration of ANN ML objects with currently available organizational learning concepts is missing.

Therefore, a research gap becomes visible here.

A synthesis of any ANN ML approach presented here, which will be enriched with process and knowledge modeling objects and required concepts and mechanisms, will support a broad acceptance of the CoNM in the corresponding research communities. Further, a synthesis ensures its broad application possibilities. A synthesis must assure that complex models do not reduce the ease of use for any realistic setting, aiming to not construct impractical modeling approaches (Curtis, Kellner, and Over, 1992).

Hence, for the construction of a CoNM, a compromise must be identified between the holism of modeling items required and practicability. In principle, a practical and holistic synthesis must address the following:

- It must look broadly on the creation of ANN modeling objects regarding the structural ANN building and task realization across systems (section 2.5.9.1) and provide modeling objects for its focus.
- It must look broadly on the creation of ANN modeling objects regarding the applicability, which includes cross-validation by various simulators (section 2.5.9.2), and provide modeling objects for its focus.
- It must look broadly on the creation of ANN modeling objects regarding the mechanisms required for their application, which includes cross-validation by different simulators (section 2.5.9.3), and provide modeling objects for its focus.
- ANN techniques must be harmonized with the understanding of organizational learning. This includes process modeling, simulation modeling, and knowledge modeling as individual learning domains.

The creation of a CoNM will, therefore, consider those points and it will be based on a practical synthesis of ANN ML approaches presented here. Considering ANN ML for the CoNM construction, certain works broaden the perspective on the use of biologically plausible mechanisms and a realistic level of detail.

2.5.10 Conclusion About Deep Learning

The conclusions drawn about DL are based on the following: first, ANN concepts are appraised; then, ANN Modeling Language approaches (MLs) follow. A DL-specific facet is then identified.

ANN Concepts: All approaches to model NN mechanisms, so-called ANN concepts, are based on a systematic way to consider knowledge in order to consider biological mechanisms at current models. The neuroscience community puts particular effort into biological plausibility, while the machine learning community puts effort into ANN capabilities. The consideration of business process perspectives, business process modeling objects, and an integration with components from a simulation modeling domain as well as a knowledge management domain have not been identified in any approach (see each individual interim conclusion). Since ANN concepts were defined in order to consider biological principles for application, for instance, at certain learning tasks which do not refer to improvements coming from a CoNM, an appropriate CoNM optimization concept, per se, is still missing. Hence, ANN concepts have neither been evaluated to be the foundation for the CoNM (as building blocks on various levels), nor have they been evaluated to evaluate elements required by an CoNM till date.

In order to decide, whether current ANN concepts can at least partly identify a CoNM foundation or which criteria is suited to be part of a CoNM criteria catalog, the following appraises criteria category-wise:

1. Since the CoNM is intended to be carried out in a holistic and general setting, any kind of *biological motivation* considered by ANN concepts is relevant since these focus on aspects which are accepted in the ANN and DL community. These serve as evolutionary optimized mechanisms, ideal to be transferred to process contexts such as the organization principle of depots in economic contexts. Criteria trying to apply accepted concepts from the ANN domain in the ANN process domain are highly relevant.
2. Since the CoNM is intended to be carried out in a holistic and general setting, any kind of *learning tasks* considered by ANN concepts is relevant since these focus

on aspects which are accepted in the ANN and DL community. These serve as application frame for biological mechanisms being suitable to be transferred to process contexts, such as the reinforced production optimization in economic contexts. Criteria trying to apply accepted concepts from the ANN domain in the ANN process domain are highly relevant.

3. Since the CoNM is intended to be carried out in a holistic and general setting, any kind of *knowledge codification* and *knowledge encodification* considered by ANN concepts is relevant since these focus on aspects which are accepted in the ANN and DL community. As these enable the access to biological mechanisms modeled by ANN, these serve as an interface for process contexts such as a company's organization structure being mapped to ANN systems. Criteria trying to apply accepted concepts from the ANN domain in the ANN process domain are highly relevant.
4. Since the CoNM is intended to be carried out in a holistic and general setting, any kind of *learning problem* considered by ANN concepts is relevant since these focus on aspects which are accepted in the ANN and DL community. As these enable the access to biological mechanisms modeled by ANN, these serve as operationalization for process optimization objectives such as the reduction of a production throughput time economic context. Criteria trying to apply accepted concepts from the ANN domain in the ANN process domain are highly relevant.
5. Since the CoNM is intended to be carried out in a holistic and general setting, any kind of *ANN basic technique* considered by ANN concepts is relevant since these focus on aspects which are accepted in the ANN and DL community. This includes individual neurons, MLPs and RNNs. As these model biological mechanisms, these techniques serve as optimization tool in process contexts. Criteria trying to apply accepted concepts from the ANN domain in the ANN process domain are highly relevant.
6. Since the CoNM is intended to be carried out in a holistic and general setting, any kind of *training approach of ANN* considered by ANN concepts is relevant since these focus on aspects which are accepted in the ANN and DL community. These serve as optimization principle and methodological proceeding for ANN concepts in process contexts. Criteria trying to apply accepted concepts from the ANN domain in the ANN process domain highly relevant.
7. Since the CoNM is intended to be carried out in a holistic and general setting, any kind of *network interpretation approach* considered by ANN concepts is relevant since these focus on aspects which are accepted in the ANN and DL community. In order to apply ANN mechanisms in process contexts, these mechanisms enable the analysis, validation, and interpretation of results. As a requirement, they allow operating in the ANN process domain. Criteria trying

- to apply accepted concepts from the ANN domain in the ANN process domain are highly relevant.
8. Since the CoNM is intended to be carried out in a holistic and general setting, any kind of *network architecture* considered by ANN concepts is relevant since these focus on aspects which are accepted in the ANN and DL community. These structures serve as structural restriction for the application of ANN techniques in process contexts. So, their structural arrangement can serve interpretation purposes. Criteria trying to apply accepted concepts from the ANN domain in the ANN process domain are highly relevant.

Since no single approach issued the use of process techniques, and concepts, which is an essential part of the ANN process domain, none of provided ANN concepts is able to at least partly stand as a CoNM foundation. Further, this means that the process optimization domain has not yet been brought together by AI techniques. Thus, a research gap becomes visible here which asks for the following: first, the construction of a CoNM because anything similar is missing; second, a CoNM ANN concept selection system that extends common criteria and is able to evaluate current ANN concepts; and third, an analysis of currently available ANN concepts for the use of CoNM relevant aspects, such as process modeling techniques, specific modeling objects, and common tool criteria.

ANN ML Approaches: Most ANN MLs are based on individual modeling objects with special *focus*, each having a unique description strength. The consideration of business process perspectives, business process modeling objects, and an integration with components from a simulation modeling domain as well as a knowledge management domain have not been identified in any approach (see each individual interim conclusion). Since ANN MLs were defined in order to describe ANN models, which do not include improvements coming from a CoNM, an appropriate CoNM ANN MLs, per se, is still missing. Hence, ANN MLs have neither been evaluated to be the foundation for the CoNM (by providing commonly accepted neuronal modeling objects), nor have they been evaluated till date to be part of a CoNM.

In order to decide whether current ANN MLs can be at least partly used as a CoNM foundation or which ANN ML item and focus is suited to be part of a CoNM evaluation catalog, the following appraises aspects category-wise: Since the CoNM is intended to be carried out in a holistic and general setting, any *modeling item* provided by ANN MLs is relevant since items focus on aspects which are accepted in the ANN modeling community. Here, one can find *ANN modeling items* referring to compartments and mechanisms, *task realization mechanisms* referring to various

devices and different units, and *computing performance criteria*. Criteria trying to apply accepted concepts from the simulation domain in the ANN process domain are highly relevant.

Since no single ANN MLs provided items or mechanisms about the use of process techniques and concepts satisfactorily, which is an essential part of the ANN process domain, none of provided ANN MLs are able to solely serve as CoNM ANN ML. Further, this means that the process optimization domain has not yet been brought together with AI techniques. Thus, a research gap becomes visible here, which asks for the following: first, the construction of a CoNM ANN ML because anything similar is missing; second, a CoNM ANN ML selection system that extends common criteria and is able to evaluate current ANN MLs to be part of a CoNM foundation; and third, an analysis of currently available ANN MLs for the use of CoNM relevant aspects, such as process modeling techniques, specific modeling objects, and common tool criteria.

Facit: Neither ANN concepts nor ANN MLs consider the integration of the process optimization domain and the ANN domain. ANN concepts and ANN MLs do not provide process, business process, or knowledge and organizational learning criteria. Here, a research gap within the ANN domain becomes visible which asks for the following: first, the construction of a CoNM taxonomy focusing on ANN concepts and ANN MLs; second, the design and demonstration of a CoNM selection and evaluation framework considering ANN concepts and ANN MLs as objects of investigation; and third, the implementation and demonstration of a CoNM optimization.

2.6 Software Tooling

The design of processes is not a one-time task and asks for a continuous optimization and adjustment of its technical foundations (see section 2.4). Hence, tools must not only focus on the design of processes but, in addition, support the update, maintenance, and analysis of process models. As numerous model experts are integrated in the process of model design, systems must provide multi-user modes and must support their different devices and operating systems, among other criteria. Since the need for training sessions can be reduced by the design of usable, intuitive, and comprehensive tools, usability issues are relevant as one out of many further criteria.

Since existing literature provides a great collection of synonyms of software tools, the following terms all are referred to as tools:

software / framework / application / program / mechanisms to apply concepts / library / environment / platform / prototype.

Since commonly-known approaches for the selection of tools intend to reflect the wisdom of the *Process Domain* and the *ANN Domain* (see Fig. 3.2), approaches for the selection of tools and contemporary criteria pools are an ideal foundation for the construction of the CoNM. Therefore, the following presents common criteria and critically appraises them with regard to the problem statement of this contribution.

Since the border between modeling and simulation tools is fluent, tools for the process model creation and simulation model creation are considered in the first sub-section. Thereafter, libraries and tools for the dealing with neuronal networks are considered in the second sub-section. This includes simulation tools for the realistic simulation of neuronal networks and tools for the training and testing of ANN in the sense of machine learning.

2.6.1 Modeling and Simulation Tools

According to the understanding of *Business Process Management* and what a BPM shall achieve, interpretations and, therefore, the expectation of tools varies. While the German language area focuses on BPM as a management concept, the English language area focuses on BPM as an automation concept for processes and workflows (Lübbe and Schnägelberger, 2016). Hence, tools from the German language area tend to focus on the design and analysis of process models (model construction, analysis, simulation, reporting, life-cycle management) and are often called *process modeling tools*, while tools from the English language area tend to focus on automation aspects (real-time data use, execution engines, rule engines, testing frameworks, event processing) and are often called *BPM suites* (Sinur and Hill, 2010; Norton, 2011; Sinur et al., 2012; Adam et al., 2013; Adam et al., 2014; Jones, Schulte, and Cantara, 2014; Dunie et al., 2015; Dunie et al., 2016; Dunie et al., 2017). Since further denominations are referred to as *Business Process Management Systems* and *Workflow Management Systems* and there exist all-in-one solutions, a lack in conceptual clarity can be identified here (Lübbe, Allweyer, and Schnägelberger, 2015). Hence, the following intends to establish a common understanding of process tools and associate this with the construction of a CoNM.

Since the construction of an CoNM refers primarily to the construction of models, management concepts shall be considered as well as relevant design and analysis aspects. Further, the use of neuronal models is intended to be integrated within the organization in order to, first, realize the full potential of constructed models through their continuous use in real-time environments, and second, modify constructed

models because of their continuous use in real-time environments. This can be interpreted to be part of an ongoing, never-ending construction process. Therefore, process modeling tools are defined to be the following:

Definition 30 (Process Modeling Tools).

The process modeling tool is referred to as an application that supports the act of modeling in accordance with the required repetitive, systematic, principal-based manner, which includes design, analysis, and automation aspects such that a model is constructed within someone's subjective (or) objective reality.

Hence, the definition demands process modeling tools to support activities which are relevant for the construction of models and includes the creation of models (their visualization and management), their use (in analyses and simulations), their monitoring (in real-time data streams), and their modification (because of optimization cycles, learning procedures, management concepts, etc.) with regard to all perspectives and EA-required fields. This excludes ERP systems and Enterprise Content-Management (ECM) because of their different focus on simple process definitions, which can be realized on behalf of a configuration of tools (Lübbe, Allweyer, and Schnägelberger, 2015). Here, the universal definition and the technical support of any process is essential.

In general, the act of selecting adequate tools for process modeling projects is connected to the use of selection frameworks and to the evaluation of its criteria. As collections of criteria are very diverse and numerous, the objective of selection frameworks cannot lie on an all-encompassing collection, but can lie on a problem-related criteria compilation (Fank, 2013, p. 4). It will focus on the most recent studies of the recent eight years. Former approaches are assumed to be considered in the establishment of those state-of-the-art approaches and, therefore, are considered implicitly.

The first sub-section therefore presents a collection of selection frameworks and common selection criteria and tries to identify CoNM relevant process modeling tool criteria. The consecutive sub-sections provide selected modeling and simulation tools, which will be considered as objects of investigation in section 4.2. Ideally, one of them can be identified to be a best candidate for the extension with CoNM concepts presented here. Then, the tools are critically appraised in the last sub-section so that a process modeling tool research gap becomes visible.

2.6.1.1 Selection of Adequate Tools

As the following approaches show, the selection of adequate process modeling and simulation tools is mostly based on the fulfillment of purpose-related criteria. Hence,

numerous frameworks and various criteria collections can be found in literature with whom the very challenging selection of adequate tools shall be simplified (Nikolai and Madey, 2009).

Criteria collections: The first category provides approaches that present, discuss, or interpret tool criteria. A regular evaluation base is not available and criteria demonstrated are only exemplary for a small set of tools. Hence, those approaches are well-suited for the identification of relevant tool criteria and the exemplary identification of tools fulfilling those criteria.

First discussions about modeling tool selections can be found in (Chrobok and Tiemeyer, 1996; Finkeißen, Forschner, and Häge, 1996; as well as Buresch, Kirmair, and Cerny, 1997).

Discussions about tool criteria for knowledge-intensive processes can be found in (Pogorzelska, 2009; Gronau, 2012; as well as Gronau and Maasdorp, 2016) and their realization mainly considers the tool called *Modelangelo*.

A collection of experience-based, project-relevant tool criteria is presented by Rosemann, Schwegmann, and Delfmann (2012).

As a new technique being used in process modeling, first discussions and criteria focus on Augmented Reality in the domain of process modeling (Grum and Gronau, 2017; Grum and Gronau, 2018b), which are considered in the tool called *Modelangelo*.

Fank (2013) presents a category-based pattern for the description of process modeling and simulation tools from which criteria can be derived. A description is presented considering six tools: *Ablauf-Profi*, *AENEIS*, *ARIS*, *Bonapart*, *GRADE-BM*, and *Nautilus*.

Criteria-based market overviews: The second category provides approaches that present, discuss, and interpret relevant tool criteria and regularly evaluate tools on the market (preferably on a wide basis). These are very suited for the identification of trend criteria and the identification of tools greatly promoted in the market.

The Business Application Research Center (BARC), Würzburg, regularly presents market overviews of diverse applications. Here, one can find overviews of tools for documentation, analysis, simulation, and process control, which are all called *process modeling tools* (Böhn, 2007; Böhn, Burkhardt, and Gantner, 2010; Böhn et al., 2014). Criteria catalogs provided here focus on common criteria, technical criteria, and application support criteria. The most current overview considers five tools (*QPR Suite 2014*, *Innovator Modeling Suite*, *Signavio Process Editor*, and *Enterprise Architect*; Böhn et al., 2014).

The Fraunhofer Institute regularly presents market overviews of modeling and simulation tools such as Drawehn, Gayer, and Schneider (2010), Herbert et al. (2011), and Drawehn et al. (2014). Here, the focus lies on functional criteria of categories such as *modeling, analysis, simulation, process execution, workflow implementation* (Drawehn, Gayer, and Schneider, 2010), their *controlling* (Drawehn et al., 2014), *interfaces, quality management*, the provision of *reference models*, and the optimization with regard to the *degree of maturity* (Herbert et al., 2011). Each market overview considers numerous tools (eight tools of Drawehn, Gayer, and Schneider, 2010, 55 tools of Herbert et al., 2011 and 53 tools of Drawehn et al., 2014).

The German Society for Process Management (Gesellschaft für Prozessmanagement—GfK) considers 26 tools in a market overview (Kuhlang, Wagner, and Moser, 2010).

The company Gartner regularly presents market studies evaluating many tools with regard to the two abstract dimensions called *ability to execute* and *completeness of vision* (Sinur and Hill, 2010; Norton, 2011; Sinur et al., 2012; Jones, Schulte, and Cantara, 2014; Dunie et al., 2015; Dunie et al., 2016; Dunie et al., 2017). The first dimension focuses on vendors being able to enable IT provider performance, while the second dimension focuses on vendors being able to convincingly articulate logical statements.

The tool market monitor of the company called BPM&O regularly presents market studies evaluating tools with regard to criteria that focus only on the design and analysis of process models. The monitor provided a tool comparison with 22 representatives in 2014 (Lübbe and Schnägelberger, 2014), with 16 tools in 2015 (Lübbe, Allweyer, and Schnägelberger, 2015), and the most current report with 29 representatives in 2016 (Lübbe and Schnägelberger, 2016) and four cloud-based tools in 2018 (Lübbe and Schnägelberger, 2018). In general, they focus on about 150 criteria of the categories *modeling, portal, reporting, process lifecycle management, collaboration, repository, controlling and monitoring*, and *simulation* (Lübbe and Schnägelberger, 2014; Lübbe and Schnägelberger, 2016). Their study on free modeling tools considered the fulfillment of more than 50 criteria of 17 free modeling tools (Lübbe, Schnägelberger, and Pelz, 2016). Cloud-based tools focused on the categories *standardization, architectures, service quality, reliability, cost models*, and *migration simplicity* (Lübbe and Schnägelberger, 2018). Tools focusing on automation aspects consider the categories *basis technologies, interfaces, deployment models, licenses, and costs* (Lübbe, Allweyer, and Schnägelberger, 2015).

Based on the BPM Suites market analysis of the Fraunhofer IESE in 2013 with 8 tools (Adam et al., 2013), 11 main categories group 43 criteria. An even greater number of 17 tools has been considered in the study in 2014 (Adam et al., 2014). Considering 115 criteria here, a two-step hierarchy of 35 sub-categories and 8 main

categories has been established. Main categories refer to *process modeling*, *process realization*, *system integration*, *process execution*, *runtime management*, *process controlling*, *BPM governance and administration* (Adam et al., 2013; Adam et al., 2014), *setup*, and *implementation basis* (Adam et al., 2013). An evaluation focused the dimensions called *powerfulness* (functions usable without configuration efforts), *comfort* (perceived quality of functions), and *functional fulfillment* (summed and weighted degree of functional fulfillment).

Identification of criteria: Since some approaches categorize criteria and present evaluations on an abstract level, for instance, Fank (2013), many approaches only show atomic criteria. Therefore, approaches are hard to compare. Further, different approaches obtain redundant criteria and provide criteria intersections. Hence, the following focuses on the identification of criteria independent from specific approaches, tries to establish a common criteria abstraction level, and appraises criteria individually such that a wide-ranging, redundancy-free perspective on tools can be constructed.

1) Tool Usage Criteria: The selection of adequate tools shall be based on the fulfillment of functional demands (Rosemann, Schwegmann, and Delfmann, 2012) which support the use of tools (Böhn, Burkhardt, and Gantner, 2010, p. 123; Böhn et al., 2014, p. 95).

- **Usability:** The tool must provide a user-friendly control interface, such that even a moderate model expert is able to create and discuss simultaneously about models in workshop sessions (Rosemann, Schwegmann, and Delfmann, 2012). This includes comfortable functions (shortcuts, drag-and-drop mechanisms, automated connect, wizards, etc.), layout optimization mechanisms (automated model creation, alignment of modeling objects, etc.), and attribute input mechanisms for objects and relations (costs, setup times, durations, maturity information, etc.) as per (Böhn, Burkhardt, and Gantner, 2010, p. 125; Herbert et al., 2011, p. 43; Gronau, 2012; Fank, 2013; as well as Lübbe, Schnägelberger, and Pelz, 2016). Further, this includes the provision of operation instructions (Böhn, Burkhardt, and Gantner, 2010, p. 125) and assistance functionality through manuals, communities, and video tutorials, among others (Lübbe and Schnägelberger, 2014; Lübbe and Schnägelberger, 2016; Lübbe, Schnägelberger, and Pelz, 2016). Especially if tools are available via cloud services, tools must be usable by simple migration efforts (Lübbe and Schnägelberger, 2018).
- **Denomination of Modeling Objects:** The tool must enable the efficient denomination of modeling objects (Pogorzelska, 2009; Gronau, 2012; Gronau and Maas-

dorp, 2016; Grum and Gronau, 2018b). As denominations are used twice, either the same entity is to be used or different indexes have to be provided automatically.

- **Management of Process Hierarchies:** The tool must enable the creation and selection of processes being part of other processes (Gronau, 2012; Fank, 2013). Ideally, the process system can be drilled over available process parts. A separate process hierarchy view then visualizes available models with regard to their hierarchical position and the management of process models (Böhn, Burkhardt, and Gantner, 2010, p. 123; Böhn et al., 2014, p. 95).
- **Management of Abstraction Levels:** The tool must enable the creation and selection of processes having different abstraction levels (Gronau, 2012; Fank, 2013). As the same process can be visualized in several variants, this is connected to a variant management (Lübbe and Schnägelberger, 2014; Lübbe and Schnägelberger, 2016). Ideally, the same process system can be drilled over available abstraction levels. A separate abstraction view then visualizes available models in regards to their abstraction levels and supports the management of process models (Böhn, Burkhardt, and Gantner, 2010, p. 123; Böhn et al., 2014, p. 95).
- **Syntax and Rule Checking:** The tool must check if the process models realized are syntactically correct and visualize design errors as they occur (Pogorzelska, 2009; Böhn, Burkhardt, and Gantner, 2010, p. 124; Gronau, 2012; Böhn et al., 2014, p. 95; Drawehn et al., 2014, p. 28). This includes the possibility to define and manage syntax and organizational rules (Adam et al., 2013; Adam et al., 2014). Since rule checking mechanisms support the correct model construction, this ensures the functioning of analysis and simulation activities.
- **Quality Management:** In addition to simple syntax checking, the tool must provide a methodological proceeding such that models are created that fulfill requirements of standards (ISO 9000 models, EFQM models). This includes the management over the entire life-cycle of models (Lübbe and Schnägelberger, 2014; Lübbe and Schnägelberger, 2016) and its tool support, for instance during the information gathering process, the automated generation of process documentations, and their optimization with regard to their degree of maturity (Herbert et al., 2011, p. 66).
- **Modeling Language Extendability:** The tool must provide a meta-model editor, such that available modeling languages can be modified and new modeling languages can be constructed (Lübbe and Schnägelberger, 2014; Lübbe and Schnägelberger, 2016). Widely accepted modeling languages are expected to be in the initial setup and must be importable (Pogorzelska, 2009; Böhn, Burkhardt, and Gantner, 2010, p. 124; Herbert et al., 2011, p. 40; Adam et al., 2013; Adam

et al., 2014; Böhn et al., 2014, p. 95; Gronau, 2012; Lübbe and Schnägelberger, 2014; (Lübbe and Schnägelberger, 2016); Lübbe, Schnägelberger, and Pelz, 2016).

- **Multi-User Support:** The tool must simultaneously consider the model creation, modification and use from different locations and several persons (Rosemann, Schwegmann, and Delfmann, 2012; Fank, 2013). This includes an easy user and right management and is essential for collaboration (Böhn, Burkhardt, and Gantner, 2010, p. 122; Böhn et al., 2014, p. 96). Further, this includes the management of conflicts because of parallel working and collaboration tools such as the positioning of notes within process models (Lübbe and Schnägelberger, 2014; Lübbe and Schnägelberger, 2016) and the number of participants as user of Workflow Management Systems (Lübbe, Allweyer, and Schnägelberger, 2015).
- **Publication:** The tool shall support an information distribution through many possible communication channels (Böhn, Burkhardt, and Gantner, 2010, p. 122; Herbert et al., 2011, p. 62; Fank, 2013; Drawehn et al., 2014, p. 28). This requires the tool to create websites which present selected models and information via the Internet (Rosemann, Schwegmann, and Delfmann, 2012) and make them public (Böhn et al., 2014, p. 96). Ideally, the information distribution enables a bidirectional information exchange. For example, portals can present models, manage access rights, and collect feedback suggestions (Lübbe and Schnägelberger, 2014; Lübbe and Schnägelberger, 2016). Further, this includes the availability of platform and tool independent viewer tools, their vector-based printing in commonly accepted file formats, such as PDF and XML (Adam et al., 2013; Adam et al., 2014; Lübbe, Schnägelberger, and Pelz, 2016), and the password protection of files (Fank, 2013) or user and role-based access (Adam et al., 2014).
- **Mulit-Language-Support:** The tool must support the most common languages (Böhn, Burkhardt, and Gantner, 2010, p. 119). Going further, this includes English, Chinese, and German (Lübbe, Schnägelberger, and Pelz, 2016).
- **Example Processes:** The tool must provide examples of processes that can simplify the use of the modeling tool and invite an exploration (Böhn et al., 2014, p. 97; Lübbe, Allweyer, and Schnägelberger, 2015).

2) Technical Concept Criteria: The selection of adequate tools shall be based on the fulfillment of functional demands (Rosemann, Schwegmann, and Delfmann, 2012) which focus on the use of technical concepts (Böhn, Burkhardt, and Gantner, 2010, p. 121; Böhn et al., 2014, p. 93).

- **Modeling Language Integration:** The tool must enable the exchange of information between different kind of models (Rosemann, Schwegmann, and Delf-

mann, 2012; Fank, 2013). This supports the prevention of redundant and inconsistent information.

- **Multi-View Concept:** As numerous modeling languages demand for perspectives, the tool must be able to manage multiple views (Pogorzelska, 2009; Herbert et al., 2011, p. 51; Rosemann, Schwegmann, and Delfmann, 2012; Gronau 2012). As a kind of definable view, a *model segmentation* enables the attribute-wise selection of modeling objects to be visualized only for a certain user group or role (Böhn, Burkhardt, and Gantner, 2010, p. 123; Böhn et al., 2014, p. 95, 96).
- **Database Integration:** For an easy creation, maintenance, and use of various models, their storage in one integrated database is essential (Rosemann, Schwegmann, and Delfmann, 2012). This further requires the management of this database and functions for the purifying, backups, model changes, and the administration (Böhn, Burkhardt, and Gantner, 2010, p. 123; Adam et al., 2013; Adam et al., 2014; Böhn et al., 2014, p. 94).
- **Workflow Implementation:** The tool must support the creation of process instances, which execute processes modeled before (Böhn, Burkhardt, and Gantner, 2010, p. 125; Drawehn, Gayer, and Schneider, 2010; Böhn et al., 2014, p. 97; Drawehn et al., 2014, p. 28) and influence each other (Adam et al., 2014; Lübbe, Allweyer, and Schnägelberger, 2015). This includes the automated notification of process participants (humans, machines, systems) in form of messages, work lists, the labeling of their current states (Herbert et al., 2011, p. 112, 139), and the administration and manipulation of process instances (Böhn, Burkhardt, and Gantner, 2010, p. 123; Adam et al., 2013; Böhn et al., 2014, p. 94; Adam et al., 2014).
- **Interface Provision:** Tools must provide interfaces which allow the integration with further technical components such as simulation specific modules, analytics components, and process monitoring and workflow management systems, among others (Böhn, Burkhardt, and Gantner, 2010, p. 121; Herbert et al., 2011, p. 107; Rosemann, Schwegmann, and Delfmann, 2012; Adam et al., 2013; Adam et al., 2014; Böhn et al., 2014, p. 94; Drawehn et al., 2014, p. 28; Lübbe, Allweyer, and Schnägelberger, 2015). Especially if tools are available via cloud services, tools must be accessible via standardized micro-service-architecture components and their use must be enabled by standardized interfaces (Lübbe, Schnägelberger, and Pelz, 2016; Lübbe and Schnägelberger, 2018).
- **Script Language:** The tool must provide a script language that simplifies the efficient modification, analysis, and simulation of models (Rosemann, Schwegmann, and Delfmann, 2012) and, therefore, supports an easy handling of different technical components. Simple scripts are intended to be created by even inexperienced users.

enced model experts. Additionally, the construction of models must be realizable by wizards and plug and play mechanisms, which are called *zero coding* and *less coding* (Lübbe, Allweyer, and Schnägelberger, 2015).

- **Version Control:** The tool must enable the management of model versions (Rosemann, Schwegmann, and Delfmann, 2012; Adam et al., 2013; Adam et al., 2014). Ideally, this is based on version control systems and repositories (Lübbe and Schnägelberger, 2014; Lübbe and Schnägelberger, 2016), and produces session protocols (Fank, 2013) for the administration (Böhn, Burkhardt, and Gantner, 2010, p. 123; Böhn et al., 2014, p. 94). This includes the methodological perspective of revision management (Herbert et al., 2011, p. 62) and supports the collaboration and enables security backups (Böhn et al., 2014, p. 94).
- **Multi-Platform Access:** The tool must be accessible by various platforms and devices (Böhn, Burkhardt, and Gantner, 2010, p. 120). This includes platforms such as Mac, Windows, Android, and devices such as desktops, smartphones, tablets, and AR-glasses. Ideally, they are accessible via the Internet (Böhn, Burkhardt, and Gantner, 2010, p. 122; Böhn et al., 2014, p. 95) and they are explicitly considered in a server-client architecture (Böhn, Burkhardt, and Gantner, 2010, p. 121; Böhn et al., 2014, p. 93).
- **Augmented Reality Integration:** The tool must provide interfaces for augmented reality hardware, software, and techniques that allow an integration with AR devices, such as AR glasses and tablets, among others (Grum and Gronau, 2017).

3) Simulation Tool Criteria: Besides process model tool specific criteria, further functional criteria about analyses, workflows, and simulations have to be considered in addition (Fank, 2013).

- **Objective Development:** The tool shall support the methodologically supported establishment of objectives for the creation of models, their analysis, and simulation on a processual level. This includes the identification and selection of process variants, benchmarking candidates, identification of strengths and weaknesses (SWOT analysis), scenario planning, parametrization of scenarios (Six Sigma), and continuous improvement (Kaizen) (Herbert et al., 2011, p. 82).
- **Process Simulation:** The tool must enable process simulations (Böhn, Burkhardt, and Gantner, 2010, p. 124; Böhn et al., 2014, p. 96) and provide common control mechanisms as simulations unfold over time. This includes the animation start, stop, pause, previous time step selection, next time step selection, and specific time step selection (Grum and Gronau, 2017; Grum and Gronau, 2018b).

- **What-if-Scenarios:** The tool must support the analysis and simulation of different scenarios (Herbert et al., 2011, p. 82; Grum and Gronau, 2017; Grum and Gronau, 2018b), which are connected to a case management (Lübbe, Allweyer, and Schnägelberger, 2015). While simple scenarios can be simulated without further simulation-specific attributes, complex scenarios require the specification of scenario-specific attributes such as specific transition probabilities, timings, and number of simulation runs, among others (Fank, 2013).
- **KPIs:** The tool must support the simulation-based creation of common key performance indicators, such as *processing time* and *processing costs* (Fröming, 2009; Herbert et al., 2011, p. 91; Adam et al., 2013; Adam et al., 2014; Drawehn et al., 2014, p. 28) and ideally allows the construction of arbitrary KPIs. KPIs shall be based on all information available in the model, and includes set-up times, roles, and costs (Fank, 2013; Lübbe and Schnägelberger, 2014; Lübbe and Schnägelberger, 2016).
- **Analyses and Pattern:** The tool must support different kinds of analyses (Böhn, Burkhardt, and Gantner, 2010, p. 124; Herbert et al., 2011, p. 104; Böhn et al., 2014, p. 96) and aim at pattern recognition (Pogorzelska, 2009; Gronau, 2012). In accordance with Fank (2013), at the very least, this includes time, amount, and cost dimensions and allows comparisons of as-is scenarios and to-be scenarios. The Fraunhofer Institute focuses on the utilization of resources and waiting queues (Drawehn et al., 2014, p. 28).
- **Reference Process Integration:** The creation, analysis, and simulation of models shall consider reference processes (Böhn, Burkhardt, and Gantner, 2010, p. 121; Böhn et al., 2014, p. 93). This includes company-specific, internal standards and external standards such as ITIL, DOMEA, CoBiT, SAP R/3, and reference models of ISO and EFQM (Herbert et al., 2011, p. 72).
- **Monitoring:** The tool must be able to receive KPIs such that real-time values can be used for analyses, simulations, and the controlling of specifications (Herbert et al., 2011, p. 107; Adam et al., 2013; Adam et al., 2014; Drawehn et al., 2014, p. 28) in real-time or off-line (Lübbe and Schnägelberger, 2014; Lübbe, Allweyer, and Schnägelberger, 2015; Lübbe and Schnägelberger, 2016). This demands for an administration of systems and data streams (Böhn, Burkhardt, and Gantner, 2010, p. 123; Böhn et al., 2014, p. 94).
- **Reports and Graphics:** As analyses and simulation runs are realized, results shall stand beside its graphical interpretation and documentation (Pogorzelska, 2009; Herbert et al., 2011, p. 48; Gronau, 2012; Drawehn et al., 2014, p. 28). Ideally, different kinds of graphics can be selected (Fank, 2013).
- **Animation of Dynamics:** The tool must provide visualizations and animations of the simulation (Fank, 2013; Grum and Gronau, 2018b). This includes differ-

ent kinds of flows: knowledge flows, data flows, and material flows (Fröming, 2009; Grum and Gronau, 2017). Further, this includes state changes and the creation of modeling objects (Grum and Gronau, 2018b).

- **Augmented Reality Integration:** The tool must provide interfaces for augmented reality hardware, software, and techniques, which allow an integration with AR devices, such as AR glasses and tablets, among others (Grum and Gronau, 2017).

4) User specific criteria: In addition to functional demands, the selection of adequate tools shall further consider company-specific characteristics (Rosemann, Schwegmann, and Delfmann, 2012). Ideally, the selected tool fits perfectly in its application environment.

- **Available Know-how:** Tools which are well-known in a company and in its project teams are to be preferred since the availability of required knowledge supports the successful realization of modeling projects (Rosemann, Schwegmann, and Delfmann, 2012).
- **Training Opportunities:** If competencies about a tool are missing, the availability of tool-specific trainings within a company are to be considered at the tool selection (Rosemann, Schwegmann, and Delfmann, 2012). This includes intern services that enable the company (Böhn, Burkhardt, and Gantner, 2010, p. 120).
- **Tool Configuration:** The tool must support the adjustment to company-specific modeling requirements (Rosemann, Schwegmann, and Delfmann, 2012). This focuses on the consideration of company-specific modeling techniques, kinds of visualizations, modeling methods, conventions (Fank, 2013), and the adjustment of reports (Herbert et al., 2011, p. 49; Adam et al., 2013; Adam et al., 2014). Further, this includes the definition of specific modeling languages with help of a meta-modeling module so that niche requirements can be met.
- **Market Segment:** The tool must consider the customer's market segment since each segment asks for different characteristics (Böhn, Burkhardt, and Gantner, 2010, p. 118).
- **Price-Performance Relation:** The tool must provide a broad functional basis for a certain company, which provides a suitable relation to the tool price (Böhn, Burkhardt, and Gantner, 2010, p. 119; Rosemann, Schwegmann, and Delfmann, 2012; Böhn et al., 2014, p. 93).

5) Vendor specific criteria: Vendor-specific criteria have to be considered (Böhn, Burkhardt, and Gantner, 2010, p. 118; Sinur and Hill, 2010; Norton, 2011; Sinur

et al., 2012; Böhn et al., 2014, p. 92; Jones, Schulte, and Cantara, 2014; Dunie et al., 2015; Dunie et al., 2016; Dunie et al., 2017).

- **Ability to execute:** The tool that comes from vendors and is able to enable IT provider performance is to be preferred. This includes criteria categories such as *product and service, overall viability, sales and execution, market responsiveness, marketing execution, customer experience, and operations* (Dunie et al., 2017).
- **Completeness of vision:** The tool that comes from vendors and is able to convincingly articulate logical statements is to be preferred. This includes criteria categories such as *market understanding, marketing strategy, sales strategy, offering strategy, business model, industry strategy, innovation, and geographic strategy* (Dunie et al., 2017).
- **Support by Experts:** The tool must be supported by the tool producer (Böhn, Burkhardt, and Gantner, 2010, p. 120,125; Rosemann, Schwemann, and Delfmann, 2012; Böhn et al., 2014, p. 92,93). Especially if tools are only accessible via the Internet and cloud services, the high service quality of vendors supports the reliability of tools, the secure provision of model data, data backups, etc. (Lübbe and Schnägelberger, 2018).
- **Reference Projects:** The tool that can visualize its capabilities in its successful use in reference projects is to be preferred (Böhn, Burkhardt, and Gantner, 2010, p. 121). Reference customers can give evidence for the functioning and provide experiences.
- **Business Model:** If tools are available via cloud services, the underlying business model the tool is provided is of importance. Costs highly differentiate if they occur per service use, per tool user, model created, or tool licenses, among others (Lübbe, Allweyer, and Schnägelberger, 2015; Lübbe and Schnägelberger, 2018).
- **Vendor Size:** It is recommended to select tools from vendors having the same size as the company of tool users because then a communication tends to be on an equal footing (Lübbe and Schnägelberger, 2014; Lübbe, Allweyer, and Schnägelberger, 2015; Lübbe and Schnägelberger, 2016).

Interim Conclusion about Evaluation Approaches: Beside the option of a new development, any tool from the process domain is principally suitable for critical appraisal of its suitability to be extended with CoNM concepts presented here. Since most tool selection approaches identify powerful tools by a best overall criteria fulfillment, these best candidates are especially suitable to be part of this CoNM evaluation. Faced with a huge number of modeling tools mentioned in previous

selection system approaches, the following intends to establish a reasoning for a preselection of tools to be evaluated in a first step, such that a suitable foundation for the CoNM can be identified in a second step:

1. Top three tools of Gartner's most current market research study (Dunie et al., 2017). This includes the *Pegasystems*, *Appian*, and *IBM* tools.
2. All tools of the Market Research Study of Fraunhofer IAO (2014) that satisfy CoNM-required dimensions such as modeling, analysis, simulation, and process execution (Drawehn et al., 2014). This includes the *ADONIS*, *ARIS*, *BIC Platform*, *SemTalk*, and *Signavio Process Editor* tools.
3. A BPM&O recommendation of cloud-based BPM tools which selected the *Business Transformation Suite* from Signavio, *Symbio Cloud*, *ARIS Cloud*, and *BPM Suit BIC* tools.
4. The tool called *Modelangelo* as a department-specific tool which considers AR techniques.
5. A reduction of this list by Prof. N. Gronau, a business process management expert, to four tools which are *ARIS*, *ADONIS*, *Modelangelo*, and *Signavio*.

2.6.1.2 ARIS Platform

The *ARIS Business Process Analysis Platform* provides a web-based platform of software tools supporting the EA levels of ARIS (see section 2.1.2.3.1). Here, a collection of tools enables the process documentation, analysis, standardization, and optimization, including their sustainable implementation and maintenance within a company's real environment. The main platform is divided in four sub-platforms which follow the life-cycle of a process management (strategy, design, implementation, and controlling) and each consists of an extensive collection of tools.

Description: The first platform, called *ARIS Strategy Platform*, supports the definition of organizational strategies, and whose realization is on behalf of process models and the continuous evaluation. This ensures the sustainable integration of process models in the company's business context. The second platform, called *ARIS Design Platform*, supports the collaborative modeling, analysis, simulation, and optimization of process models. This refers to models required by the *House of ARIS* (see section 2.1.3.2) and includes their publication as well as the management of ISA to be modeled. The third platform, called *ARIS Implementation Platform*, supports the workflow-based realization of process models designed and enables service-oriented process execution. This ensures the effective and efficient application of models in the company's operations. The fourth platform, called *ARIS Controlling Platform*, supports the monitoring of current process models and their

controlling on behalf of company-wide compliance management concepts. This ensures the effective and efficient realization of models constructed.

Side facts: First conceptual prototypes of the software tool were developed by Prof. August-Wilhelm Scheer of the *Institute for Business Informatics* of the University of Saarland (Keller, Nüttgens, and Scheer, 1992). A first version was developed by the *IDS Scheer Consulting GmbH* in 1992 and was known under the name *ARIS Toolset*, which was then published under commercial software license (The Scheer Group, 2019). After its first software releases, *Software AG* overtook the development in 2009 and, since then, various major releases have been published. The most current version is the *ARIS Platform* of vers. 9 (The ARIS Community, 2019).

Example: The simple appearance of a very complex integration of various tools of the ARIS Platform can be seen in Fig. 2.77. It shows the simulation of a process model on behalf of the tool called *ARIS Architect* while numerous steps before have been realized on behalf of other tools.

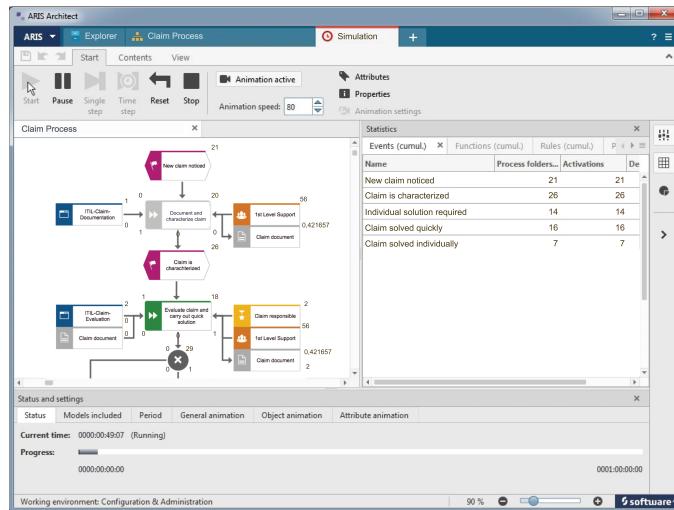


Figure 2.77 Example of the Simulation Run Realization on Behalf of the ARIS Platform (created to show the strength of the object of investigation)

The process model shown here refers to an example which has been presented in Fig. 2.11. While this process model has been designed by simple drag-and-drop mechanisms and probabilities have been assigned beforehand, in the figure, one can see the animation of a simulation run. On the left, currently activated modeling objects are highlighted within the process model. On the right, KPIs such as the activation for specific events, functions, rules, and so on, are either visualized on a cumulative or on an individual basis. The bottom shows meta-simulation data, such as the current simulation time or its progress, and the top provides an access for simulation controlling mechanisms. This refers to a start, pause, and stop button. As can be seen with this example, the comfortable process model construction and the realization of simple simulation runs can be carried out intuitively.

Critical appraisal: The strengths of the ARIS Platform refer to the collaborative process model construction, their integration with business objectives, and their operational realization. Since common modeling languages, such as the EPC, eEPC, and BPMN are supported by the tool, modeling language criteria from the business process domain are satisfied.

The tool even provides powerful wizards to transform models so that the same model can be expressed by different modeling languages. The use of repositories particularly assures the consistency of models so that any change is reflected in affected models. Further, the definition of modeling conventions and analyses checking various structure rules are enabled by the modeling tool. Therefore, errors can be displayed right within the process model so that any kind of model can be efficiently corrected. Hence, common tool criteria focusing on the modeling and simulation are satisfied as well.

Additionally, The specification of simple simulation runs can be carried out on behalf of the tool. Here, wizards provide efficient mechanisms to systematically collect required simulation parameters. Animations then visualize the current state of simulation runs on a 2D base. Regrettably, the possibility to specify the simulation model graphically must be limited to the assignment of probabilities within the process model and the aforementioned wizards. So, more complex simulation runs cannot be realized and a flexibility, such as that enabled by programming libraries, cannot be realized. Further, neither 3D animations nor the use of AR techniques showing process animations within real-world augmentations are provided. Therefore, the modeling language criteria coming from a simulation domain cannot be satisfied completely.

The provision of modeling elements required by an adequate knowledge management has not been realized to a full extent, although missing parts simply can be extended by configurations. This is accomplished by powerful modeling conven-

tion and syntax definition wizards simplifying the extension with further modeling objects.

As no ANN modeling object is provided, modeling language criteria from the NN and ANN domains cannot be satisfied. Since the generic ARIS meta-model only considers business processes (see Fig. 2.9), an adequate implementation requires high effort.

Despite the presence of a great collection of tools as a platform, machine learning libraries or simulation engines required for the simulation of ANN are not provided at all. Faced with the House of ARIS (section 2.6.1.2) and its powerful EA concepts (section 2.1.2.3.1), an adequate infrastructure can be identified with the ARIS platform for a powerful integration opportunity. Regrettably, due to the limitation by a commercial license, the tool is not suited for implementations realizing a CoNM, as focused on in this contribution.

2.6.1.3 ADONIS

The *ADONIS New Platform* provides a web-based platform of software tools supporting the collaborative process model construction, their analysis, simulation, and optimization. As it allows the customer-specific definition of meta-models, it can be seen as object-oriented meta-model tool which adjusts by simple configurations to its modeling objectives (Junginger et al., 2000, p. 393). As is defined by the tool's original architecture, the user only accesses the tool via an *user interface* (first level), so that a collection of independent *application components* can be used (second level). These include components for survey, modeling, analysis, simulation, evaluation, external components, document generation, process cost calculation, transformation, import, and export (Junginger et al., 2000, p. 394). Components all are built on the *core* of the modeling system which integrates the modeling environment and stores information in repositories. Both refer to the third tool at the very last architecture level.

Description: The modeling is carried out in a drag-and-drop manner as is usual for process modeling tools. Here, process models can be visualized in three ways. The models are shown as a *graphics*, they are provided in form of a *table*, or as a structured flow *text*. Since a dashboard enables the organization of models constructed by folders and process models can be related to sub-process models, a hierarchy of process models with an arbitrary number of levels can be easily built. Further, powerful validation mechanisms support the syntactic correct model construction. Since quality concepts are considered by the tool, a stage-wise model construction is enabled. So, models start to evolve on a *draft* level. Then, they are validated by a pool of stakeholders and follow a proper life cycle. Attributes such as

risk information or probabilities can be assigned for each modeling object. So, an extensive number of predefined analyses or simple simulation runs can be realized. Since as-is processes can be compared with to-be processes, measurements for an optimization can be derived and visualized graphically.

Side facts: First conceptual prototypes of the software tool were developed by Prof. Dimitris Karagiannis of the *BPMS Group* of the Institute of Informatics and Business Informatics at the University of Vienna (Drawehn et al., 2014). After its first software releases, the *BOC Information Technologies Consulting AG* was founded and overtook the development in 1995. This was published under commercial software license (The BOC Group, 2019) in three variants: the *Starter Edition*, the *Enterprise Edition*, and the *Community Edition*. Since the latter edition is available as freeware providing a limited functionality, and the first edition is available under a reduced license providing a limited functionality as well, the following focuses on the Enterprise Edition. To date, various major releases have been published and the most current version is the *ADONIS NP* of vers. 6.0.

Example: The simple model construction on behalf of the ADONIS Platform NP can be seen in Fig. 2.78. It shows the construction of an example application process.

In the figure, one can find the working space in the center providing a BPMN model of the application process. The overview of the current view is realized on behalf of a separate window, which is placed at the bottom. On the left, one can find the hierarchy of models constructed. Further functions are placed on its side. On the right, one can see the mechanisms to include stakeholders commenting on the current process model.

The figure visualizes the comfortable model construction on behalf of the tool. Since an effective management of rights can be realized per model, *designers* are able to modify models stored within the repository and *readers* are able to give comments. So, the collaborative model construction is realized in the sense of ADONIS. Although various persons can be included in the process construction, a collaborative working on the same model in real time is not supported.

Critical appraisal: The strengths of the ADONIS NP Platform refer to the process model's construction, analysis, and collaborative validation. Issues about their simulation and optimization are partly reflected. Since common modeling languages such as the BPMN are supported by the tool, modeling language criteria from the business process domain are satisfied.

Since the tool builds on modeling object repositories, the consistency of models is supported: any change is reflected in affected models. The syntax is checked

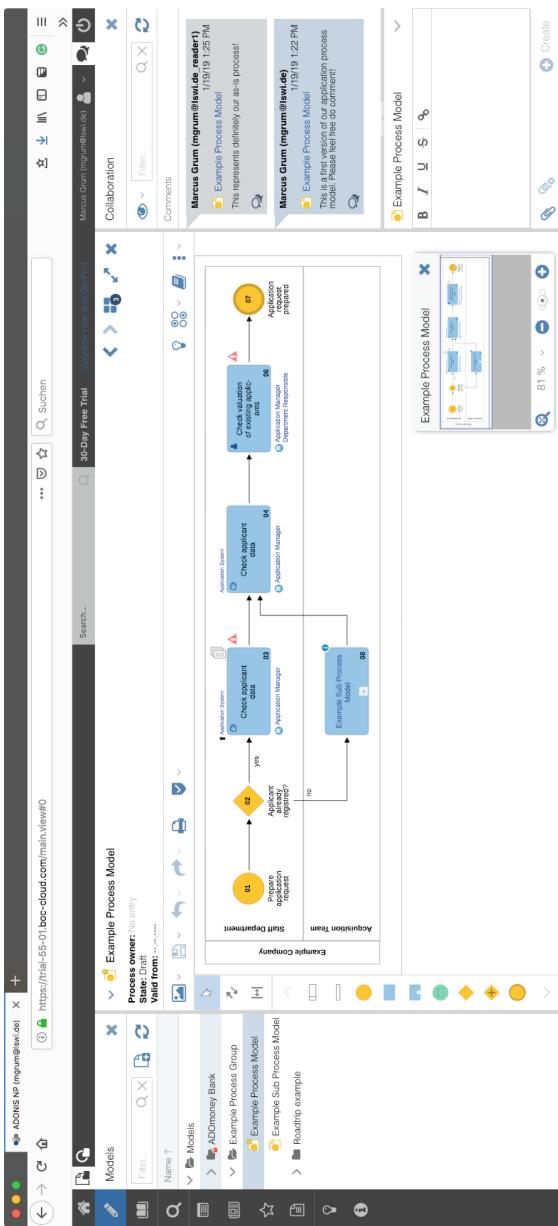


Figure 2.78 Example of ADONIS (created to show the strength of the object of investigation)

automatically by the tool and recommendations reflecting best practices, methodical proceeding steps, and process architecture requirements are given. Errors, warnings, or recommendations then directly link right within the process model, so that it can be efficiently corrected. Hence, common tool criteria of process modeling tools are satisfied.

As standard attributes and parameters can be assigned for any modeling object, simple simulation runs can be realized. Here, wizards provide efficient mechanisms to systematically collect required simulation parameters. Regrettably, the possibility to specify the simulation model graphically must be limited to the assignment of probabilities within the process model and the aforementioned wizards. So, more complex simulation runs cannot be realized and a flexibility, such as that enabled by programming libraries, cannot be realized. Further, neither 3D animations nor the use of AR techniques showing process animations within real-world augmentations are provided. Therefore, the modeling language criteria coming from a simulation domain and simulation tool criteria cannot be completely satisfied.

Further, modeling elements required by an adequate knowledge management are not provided. However, these can be extended by simple configuration efforts. As no ANN modeling object is provided, the corresponding modeling language criteria cannot be satisfied. An adequate implementation would require high effort.

Although the platform integrates a great number of modeling concepts, machine learning libraries or simulation engines required for the simulation of ANN are not provided at all. Due to the limitation of a commercial license, the tool is not suited for implementations realizing a CoNM, as focused on in this contribution.

2.6.1.4 Modelangelo

Modelangelo is a non-proprietary tool for the modeling and analysis of business process models. While common process modeling languages are supported by this tool, focus lies on knowledge-intensive business processes (Gronau, 2012).

Description: The modeling is realized in a drag-and-drop manner. As meta-models of process modeling languages can be specified by tool users, modeling entities can be visualized corresponding to related views and an automated live-syntax validation can be carried out. The latter ensures the convention-specific model creation which is essential for simulations. During the modeling, arbitrary attributes can be assigned for each modeling item which builds the foundation for analyses.

The tool provides a great collection of different kinds of analyses. Examples can be found in the following: *relevancy report*, *information occurrence pattern*, and *conversion report* (Fröming, 2009). As modeling items are stored in a local database, users can import further analyses on base of SQL syntax as well.

Side facts: A first version of the tool presented was called *K-Modeler*. This version was developed by Prof. Dr.-Ing. Gronau and the department of *Business Informatics and Electronic Government* in Potsdam (Gronau, 2006a). It served as a blueprint for a tool called *Modelangelo* (Gronau, 2009), which is now developed by the department of *Business Informatics, esp. Processes and Systems*. Since then, various releases have been published and the most current version is the *Modelangelo* of vers. 2.1.0 (KMDL Blog, 2019). Although, this is not published under an Open Source license, the codebase can be accessed because of the support of Prof. Dr.-Ing. Gronau.

Example: An example for the model creation with Modelangelo can be seen in Fig. 2.79. It refers to an open innovation process model of Jenni and Ziltener (2007) specified on behalf of the KMDL.

In the figure, one can find the modeling working space in the center of the tool. Since the activity view was chosen, available modeling items can be found within the *palette* on the left or within the selection of the most frequently used items at the top of the tool. *Properties* and *attributes* are to be specified on behalf of wizards, which can be found on the bottom and on the right of the tool. Further, a two-level hierarchy of process models can be accessed via the *project explorer* on the right. Here, the first level refers to the projects and the second level to its process models. The overview of the working space is accessible via the *outline* on the bottom left of the tool.

As can be seen with this example, the modeling can be carried out intuitively and the specification of parameters for analyses is straightforward. As the tool provides modeling information via a database and wizards allow the specification of SQL-based statements, a flexible analysis and model specification is realized.

Critical appraisal: Modelangelo strengths include the support of the business process model construction which allows further analysis. Since common business process modeling languages are supported, for instance, EPK, eEPK KMDL and PMDL, a flexible and efficient use of modeling items is realized. Hence, common process modeling language criteria are satisfied. The tool particularly shows strengths in the flexible specification of parameters and analyses.

Regrettably, a simulation engine is not provided at all. Hence, modeling language criteria from the simulation domain are not satisfied. Since first attempts deal with the simulation in Modelangelo and the integration of AR techniques in the modeling and simulation process, which refer to Grum and Gronau (2017) as well as Grum and Gronau (2018b), the tool is suited for corresponding implementations.

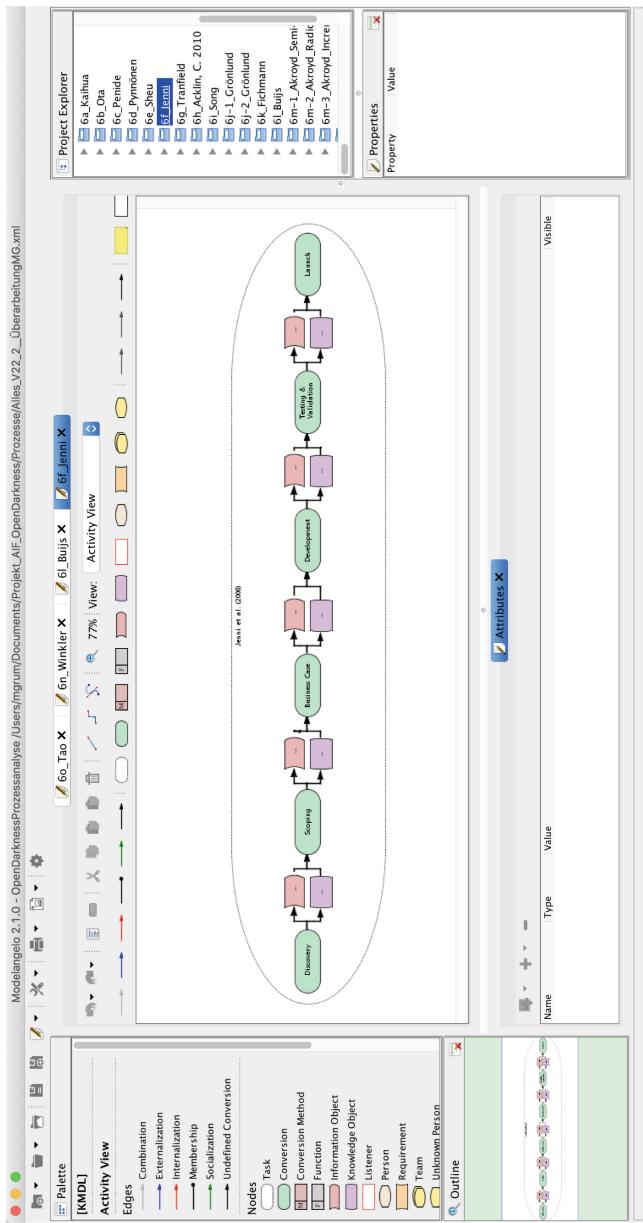


Figure 2.79 Example of Modelangelo (created to show the strength of the object of investigation)

Because of the tool's focus on business process models and knowledge intensive modeling concepts, commonly-known knowledge management approaches are considered in the tool and corresponding criteria can be satisfied.

As neither NN and ANN modeling entities are available as modeling concepts in the tool nor machine learning approaches are provided by the tool, neuronal modeling language criteria cannot be satisfied. As the tool is accessible on a code basis and only in combination with further simulation engines and ANN tools, dissatisfied criteria can be addressed and the Modelangelo can be suitable for implementations carrying concepts of the CoNM.

2.6.1.5 Signavio Process Manager

The Signavio Process Manager is a proprietary web-based tool for the collaborative modeling, analysis, and simulation of business process models. Its part of the Signavio's *Business Transformation Suite* that further includes the *Workflow Accelerator*, which allows the realization of workflows on base of process models specified, the *Process Intelligence*, which allows the analysis of them, and the *Collaboration Hub*, which allows the collaborative working on corresponding models (Signavio Team, 2017). Since the CoNM intends to construct and simulate process models, the following focuses on the *Process Manager* and considers its interplay with the other parts of the Business Transformation Suite.

Description: The modeling can be carried out efficiently with the Process Manager since common modeling languages shapes are provided and drag-and-drop mechanisms allow the comfortable model specification (Signavio Team, 2015). Hierarchies of processes and sub-processes can be constructed and linked directly within the process models. In analyses and simulations, the entire hierarchy of models is then considered. Several powerful syntax checking mechanisms allow the error identification for all kind of models. Here, even the consideration of customer-specific modeling conventions can be realized. Each kind of error or warning is directly visualized within the corresponding process model. Further, modeling is enabled because of the possibility to create *views*. So, for each view specified, various kind of modeling entities shall be visualized.

Several analysis wizards focus on the process models specified, and examples can be found in the following: *process cost analysis*, *resource consumption analysis*, and *risks and control report*. For each kind of analysis, relevant attributes can be specified for any modeling item within the modeling editor. Results can then be downloaded as an Excel file.

Further, process models can be considered in simulations. Here, predefined result plots, such as histograms, times for task realization, and interactions, simplify the

simulation run realization. Relevant variables can be specified again for any modeling item within the modeling editor. These include probabilities distributions (normal, uniform or exponential distributions), frequencies and durations (Clauberg and Thomas, 2017).

Side facts: The first version of the Process Manager was developed by a team of alumnae from the *Hasso Plattner Institute* in Potsdam (Decker, Overdick, and Weske, 2008). This was under the *Oryx project*, which was published under an Open Source license, and served as a blueprint for the *Signavio Process Manager* (Kunze and Weske, 2010). After its first software releases, the *Signavio GmbH* was founded in 2009 and, since then, twelve major releases have been published of which the most current version is the *Process Manger* of vers. 12.9.2 (Signavio GmbH, 2019).

Example: The simplicity of its use can be seen in Fig. 2.80. It shows a process simulation example consisting of four tasks of a claim process.

While the process model has been specified and simulation parameters have been set before in the editor, in the figure, one can see the simulation wizard of BPMN diagrams. Alternatively, DMN diagrams can be simulated as well. Here, the current state of the simulation is shown in blue within the process model itself. This refers to a decision being made by *Department A* about the fulfillment of requirements of a customer claim. The bottom shows the collection of simulation scenarios and allows the modification of relevant parameters. Simulation runs can be started on behalf of control mechanisms displayed in the center of the figure. They further allow the choice of the scenario to be simulated, to pause and step through the simulation, to restart, and to loop the current simulation run. Simulation results are visualized on the right of the figure and refer to costs, total cycle times, and times for the resource consumption. More detailed results and analyses can be downloaded as Excel files.

As can be seen with this example, simulation wizards greatly simplify the simulation specification. Simulation results can quickly be produced and animated within the process model. On the other side, as simulation and process models become more complex and further, non-standard KPIs shall be considered and the wizards seem to be a burden rather than a flexible tool. So, the interplay has to be critically appraised.

Critical appraisal: The Signavio Process Editor has its strengths in form of the support of the business process model construction and allows their further analysis and simulation. Here, common business process modeling languages are supported such that a flexible use and efficient dealing with business process modeling shapes is realized. Hence, common process modeling language criteria are satisfied; even

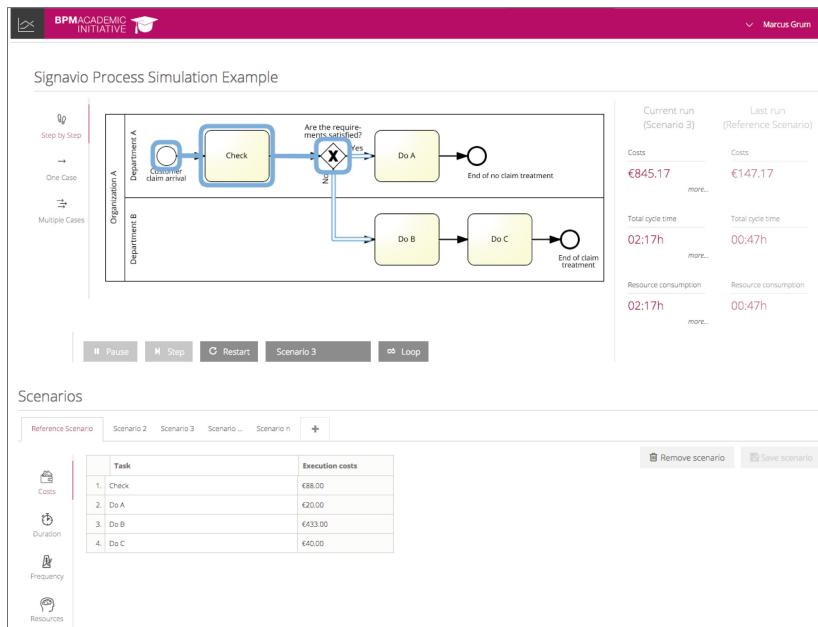


Figure 2.80 Process Simulating with Signavio (created to show the strength of the object of investigation)

the dealing with the arrangement of process models with a goal-oriented modeling is addressed.

Although the tool enables process simulations, a graphical simulation model specification equivalent to the process model construction is not supported at all. The simulation model is only accessible by a collection of simulation wizards. So, modeling language criteria from the simulation domain are not satisfied, but the tool is suited for corresponding implementations.

Because of the tool's focus on business process models, no modeling concepts coming from the knowledge management domain are considered, and ANN modeling entities are not provided at all. So, corresponding modeling language criteria are neither satisfied nor suited for efficient implementations.

The Signavio Process Manager is comfortable to use as a process modeling tool; easy drag-and-drop mechanism and a great collection of wizards and syntax checking tools simplify the model construction. The possibility to animate process

simulation runs (Fig. 2.80) is particularly suited to give insights in the internal functioning of process models as it is a requirement for the construction of a CoNM. Regrettably, AR techniques are not considered in the modeling phase nor in simulation contexts. Although the tool allows the specification of a collection of scenarios, benchmarks of process simulations are limited to two simulation runs. Further, the building of a hierarchy of scenarios and parts of scenarios is not enabled. Thus, a flexible reuse of simulation components cannot be realized at all. So, the simulation specification is limited to the use of simulation settings provided by wizards, rather than the enabling of a flexible and powerful simulation model specification, as realized by programming libraries.

Further, despite all of the strengths of the Signavio Process Manager, machine learning algorithms and ANN concepts are not provided by the tool, making it unsuitable for easy implementations carrying concepts of the CoNM. This argument becomes further clarified as the tool is proprietary so that the codebase for the first is not accessible and that of the second is not allowed to be modified as its not provided on a Open Source license.

2.6.1.6 AnyLogic

AnyLogic is a proprietary simulation tool which enables system dynamic, event-based, and agent-based simulation as well as their combination in multi method simulations (Grigoryev, 2018). Its focus lies on the general realization of simulations such that a specialization on ANN or business process contexts cannot be identified.

Description: The core of AnyLogic is a graphical simulation language wherein simulations can be designed on behalf of the following four kinds of commonly known visualizations: stock and flow diagrams, state charts, action charts, and process flowcharts. More complex models, that cannot be designed on behalf of graphical visualizations, can be specified on behalf of custom-made, Java-based code extensions. Industry-specific libraries provide further characteristic modeling entities and enable the construction of simulation models. Examples can be found in the *Rail Library*, which brings in rail models, deposits, and terminal stations, the *Pedestrian Library*, which brings in walker models for airports and shopping malls, and the *Process Modeling Library*, which brings in real-world systems, processes, and resources. Particularly, 2D and 3D interactive animations of simulation runs are enabled (Jung and Hoehe, 2015; Zambrano et al., 2016).

Side facts: The first version of AnyLogic was developed by the *Distributed Computer network* research group at Saint Petersburg Polytechnic University. It was called *COVERS*, which stands for **C**Oncurrent **V**ERification and **S**imulation (Bor-

shchev, Karpov, and Roudakov, 1997). After its first three software releases, *The AnyLogic Company* was founded and *AnyLogic 4* was released in 2000. Its new name was derived from the idea that any simulation method can be combined using one model. Since then, four major releases have been published and the most current version is *AnyLogic* of vers. 8.3.3 (The AnyLogic Company, 2019).

Example: An example simulation run is visualized in Fig. 2.81. It is based on the *Pedestrian Library* and part of the *How-To Example Models*.

In the figure, one can see AnyLogic's main window in the background. Within this window, the project structure is on the left. Properties of modeling entities can be found on the right of this window and parts of the simulation model are visualized in its center. Here, the spatial arrangement is designed which consists of a combination of various rectangles, such as building walls highlighted in orange, a waiting space highlighted in green, and three service points having serpentine waiting queues each. The entrance can be found on the left of the building and its point of departure is situated on the right. As the simulation run is started, a proper window appears which can be found in the figure foreground on the right. By selection, the simulation is animated either in 3D or 2D. The example is further clarified by explanations about the underlying simulation models from the *Space Markup Language*-based view (section 2.3.3.1) and the corresponding view on its *Flowcharts* (Fig. 2.40), which links spatial elements by process descriptions. As can be seen with this example, the design of simulations is very comfortable. Particularly, the animations are ideal for interpretations because each walker's way can be followed up like in a movie.

Critical appraisal: AnyLogic clearly shows its strengths in the specification of simulations and their animation. Since commonly used diagrams (stock and flow diagrams, state charts, action charts, and process flowcharts) are used for the simulation specification, the very basis modeling entities such as control nodes, process steps, and relations are available. Specific modeling entities that are required for business process modeling, knowledge management, or ANN modeling are not provided at all. So, only the very simple modeling criteria can be satisfied.

Although AnyLogic creates simulations which are animation ready, AR techniques are not used for animations. Further, machine learning mechanisms or biologically realistic ANN models are not considered at all. Thus, the corresponding evaluation criteria cannot be satisfied either.

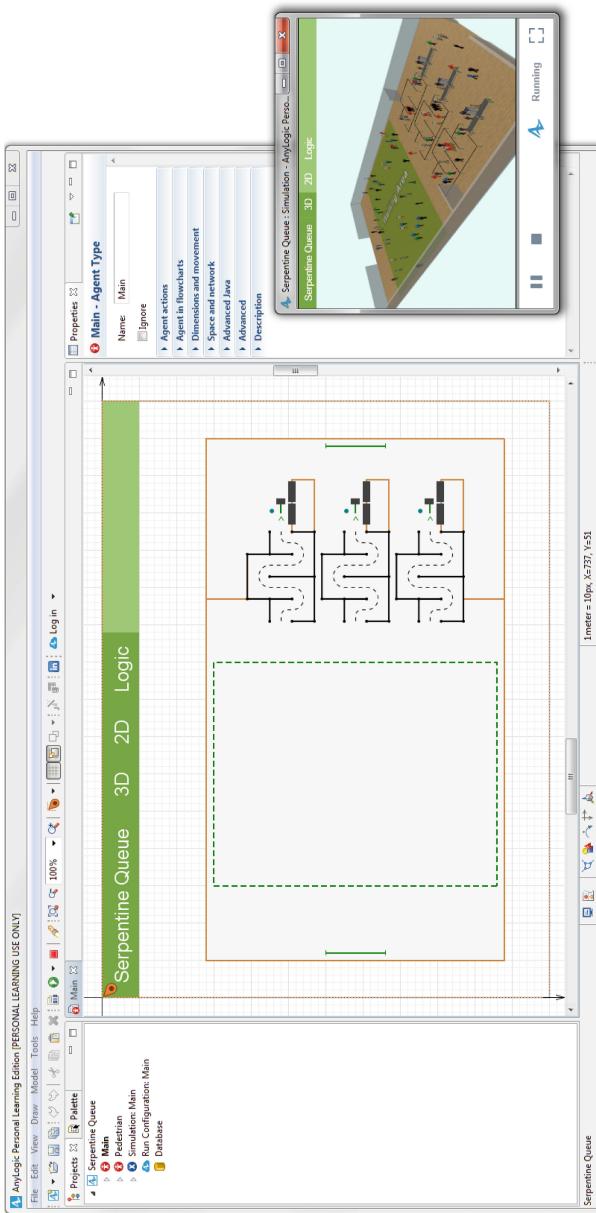


Figure 2.81 Serpentine Queue Example of AnyLogic (created to show the strength of the object of investigation)

2.6.1.7 SimPy

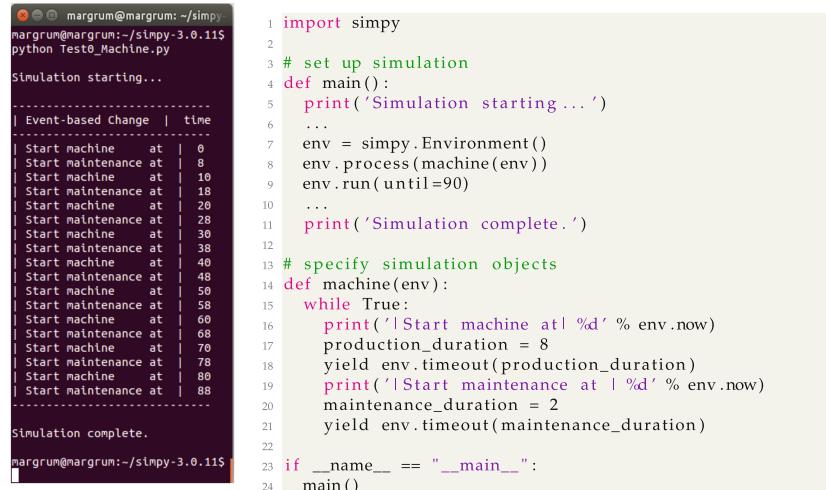
SimPy is a simulation tool, which enables process-based discrete-event simulation using the programming language called *Python* (Matloff, 2008). Its focus lies on the general realization of simulations so that a specialization on ANN or business process contexts cannot be identified.

Description: SimPy's simulation core refers to a *simulation environment* that notices events generated by *processes* and manages their interplay and shared resources. It enables a monitoring of individual processes and their resource usage and traces events of the entire simulation (Muller and Vignaux, 2003; Matloff, 2008). Further, it ensures that simulations are either realized as fast as possible or synchronous to the wall-clock time so that *real-time simulations* can be carried out. As a simulation base, various layers have evolved that can provide manufacturing objects, such as *ManPy* (Dagkakis et al., 2013).

Side facts: SimPy was initially developed by Klaus G. Müller and Tony Vignaux in 2002 and besides its community, O. Lünsdorf, S. Scherfke, and K. Turner joined the development team (Team SimPy, 2019a). Now, its available under the MIT Software license and its most current version refers to *SimPy* of vers. 3.0.11 (Team SimPy, 2019b).

Example: An example simulation run is visualized in Fig. 2.82. It shows the process simulation of a production having only one machine. Since this machine needs to be maintained two hours after having run for eight hours, the question of what the undisturbed schedule of the production looks like arises.

On the left, one can find the console-based simulation result. The section to right of the figure shows the corresponding programming code which is further explained. While the use of the SimPy library is declared in the first line of code and lines #23–#24 instruct the Python interpreter to start interpreting the main simulation and the corresponding function is specified from lines #4–#11. For this, first, the simulation environment is defined (line #7). Then, the process of the machine is declared. Finally, the simulation of 90 discrete time steps is started (line #9). The machine process is specified from lines #14–#21: As long as the simulation runs (line #15), either the production process step (line #18) or the maintenance process step is realized (line #21). The duration of each process step is specified in line #17 and line #20. The simple monitoring output, which can be found on the left of the listing, is created on behalf of console printouts with lines #5, #11, #16, and #19. As can be seen with this example, the codebase is well-suited to precisely specify simulation models, but the definition of complex process models is not comfortable at all because a graphical visualization of the model is missing.



```

margrum@margrum:~/simpy
margrum@margrum:~/simpy-3.0.11$ python Test0_Machine.py
Simulation starting...
-----
| Event-based Change | time
-----
Start machine at 0
Start maintenance at 8
Start machine at 16
Start maintenance at 24
Start machine at 28
Start maintenance at 32
Start machine at 36
Start maintenance at 38
Start machine at 40
Start maintenance at 48
Start machine at 56
Start maintenance at 58
Start machine at 66
Start maintenance at 68
Start machine at 76
Start maintenance at 78
Start machine at 80
Start maintenance at 88
-----
Simulation complete.

margrum@margrum:~/simpy-3.0.11$
```

```

1 import simpy
2
3 # set up simulation
4 def main():
5     print('Simulation starting ...')
6     ...
7     env = simpy.Environment()
8     env.process(machine(env))
9     env.run(until=90)
10    ...
11    print('Simulation complete.')
12
13 # specify simulation objects
14 def machine(env):
15     while True:
16         print('!Start machine at %d' % env.now)
17         production_duration = 8
18         yield env.timeout(production_duration)
19         print('!Start maintenance at %d' % env.now)
20         maintenance_duration = 2
21         yield env.timeout(maintenance_duration)
22
23 if __name__ == "__main__":
24     main()
```

LISTING 2.3: SimPy Simulation Code Example.

Figure 2.82 Code Example for a Simulation with SimPy (created to show the strength of the object of investigation)

Critical appraisal: SimPy's strengths lie in its code-based model specification of simulation runs, which are on basis of events. Although it is theoretically possible to carry out continuous simulations with SimPy, the required mechanisms need to be implemented before. Further, simulations can be carried out having a fixed step size, but this stresses processors and overkills SimPy.

As SimPy has no graphical user interface at all, no process modeling shape is provided and common modeling language criteria cannot be satisfied at all. Regrettably so, ANN models cannot be designed and simulated as well.

Since neither algorithms for ANN simulations, nor for machine learning approaches are provided, the ANN tool criteria cannot be satisfied and the intention to realize complex ANN simulations requires high implementation efforts.

2.6.1.8 MATLAB

MATLAB is a proprietary tool that focuses on the solution of mathematical problems and their visualization (Angermann et al., 2014). Its focus lies on the use of matrices and is the reason behind its name, which stands for **MAT**rix **L**ABoratory. Although a great collection of wizards simplify dealing with MATLAB on a graphical base,

its foundation is a programming language which can be interpreted on desktops, clusters, and even embedded systems and can be executed on behalf of processors and graphical processing units (Bergstra et al., 2010).

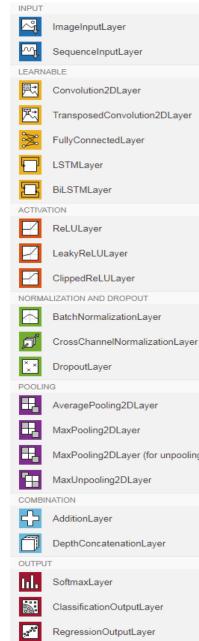
Description: As small *scripts* containing functions or entire programs can be encapsulated in libraries, which MATLAB calls *toolboxes*, their enrichment with state-of-the-art approaches by the community and their exchange is supported. MATLAB offers a powerful and highly specialized collection of proprietary toolboxes and side-products including *Simulink*, which enables time-based simulations, *Stateflow*, which is used for event-based simulations, and *Deep Learning Toolbox*, which formerly known as *Neural Network Toolbox* (Demuth and Beale, 1993) and enables the dealing with ANN (Kim, 2017).

Side facts: MATLAB was initially developed by Cleve Molder at the University of New Mexico in 1974 (Angermann et al., 2014). Since the foundation of the company called *The Mathworks* of Natick, Massachusetts, USA in 1984, MATLAB releases can be found on a yearly basis (MathWorks, 2019). The most current version is *MATLAB* of vers. 9.5 (*R2018b*).

Example I: A demonstration of MATLAB's Deep Learning Toolbox is visualized in Fig. 2.83. On the left, one can see layers and modeling shapes, which are provided by the Deep Learning Toolbox and its graphical visualizations of network structures. In order to give an impression of the use of these visualizations, an example network has been specified in the listing on the right. Thereafter, Its visualization is considered.

Here, a network is built that consists of nine different layers, each having various neurons. The basic structure is created on behalf of routine procedures so that different kind of layers simply can be named (lines #2–#11) and the network can be built in one line (line #12). Individual network structures are created thereafter. A convolutional layer is defined which intends to skip the fourth and fifth layer (lines #15–#18). The network structure, in total, then can be checked with help of the wizard called *Deep Network Analyzer* (line #21). As lines #10 and #18 are commented out in the listing and, therefore, the network is incomplete, this wizard helps to correct the implementation (Fig. 2.84).

In Fig. 2.84 (a), one can find the automated error analysis on behalf of the wizard called *Deep Network Analyzer*. Three errors have been identified that can be referred to the lines commented out in Listing 2.4. First, the layer called *skippedConvolutionLayer_4* provides an output, which is not considered in the network. Second, the layer called *AdditionLayer_6* is incomplete because of a missing input. Finally,



```

1 % # create network structure with routines #
2 layers = [
3     imageInputLayer([32 32 3], 'Name', 'InputLayer_1')
4     convolution2dLayer(5,16, 'Padding', 'same', 'Name',
5         'ConvolutionLayer_2')
6     reluLayer('Name', 'RectifiedLinearUnitLayer_3')
7     convolution2dLayer(3,16, 'Padding', 'same', 'Stride', 1, 'Name',
8         'ConvolutionLayer_4')
9     reluLayer('Name', 'RectifiedLinearUnitLayer_5')
10    additionLayer(2, 'Name', 'AdditionLayer_6')
11    fullyConnectedLayer(10, 'Name', 'FullConnectionLayer_7')
12    % softmaxLayer('Name', 'SoftmaxLayer_8');
13    classificationLayer('Name', 'OutputLayer_9')];
14
15 % # add individual network structures #
16 skippedConvolutionLayer_4 = convolution2dLayer(3,16, 'Padding',
17     'same', 'Stride', 1, 'Name', 'skippedConvolutionLayer_4');
18 net = addLayers(net, skippedConvolutionLayer_4);
19 net = connectLayers(net, 'RectifiedLinearUnitLayer_3',
20     'skippedConvolutionLayer_4');
21 % net = connectLayers(net, 'skippedConvolutionLayer_4',
22 %     'AdditionLayer_6/in2');
23
24 % # check net with Deep Network Analyzer #
25 analyzeNetwork(net)

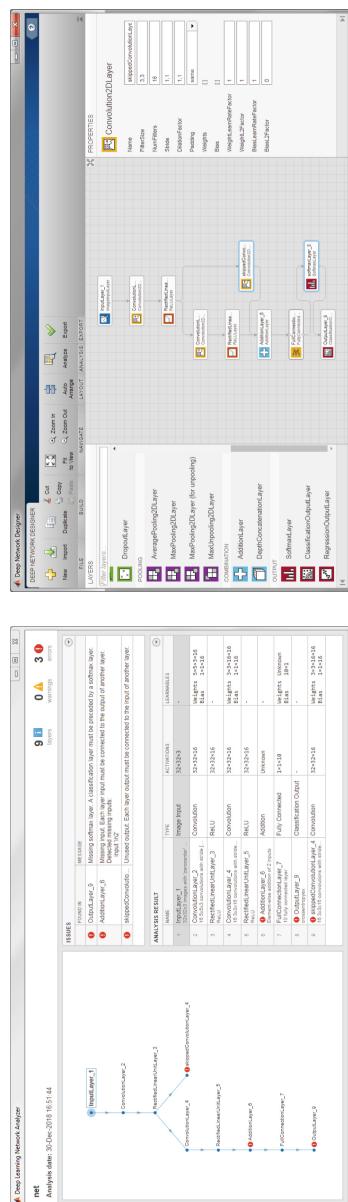
```

LISTING 2.4: MATLAB Deep Learning Toolbox Code Example.

Figure 2.83 Code Example for a Deep Network with MATLAB (created to show the strength of the object of investigation)

the layer called *OutputLayer_9* is faced with a non-optimal input and demands a softmax layer output. In Fig. 2.84 (b), one can find the network visualization on behalf of the wizard called *Deep Network Designer*. In a drag-and-drop manner, individual layers can be considered in the network construction and then they can be connected by drawing relations with the mouse. So the missing softmax layer and the missing connections from the *skippedConvolutionLayer_4* to *AdditionLayer_6* have been added manually. The corresponding modeling shapes have been highlighted in blue.

Since MATLAB knows the open format standard to represent deep learning models called *ONNX*, developers are able to easily exchange ANN models between MATLAB and state-of-the-art tools such as *TensorFlow* and *Caffe2* (ONNX-Community, 2018). Further professional wizards focus on the the training and testing of ANN,



(a) Network analysis and automated error identification (Deep Network Analyzer).
 (b) Interactive network construction and error correction (Deep Network Designer).

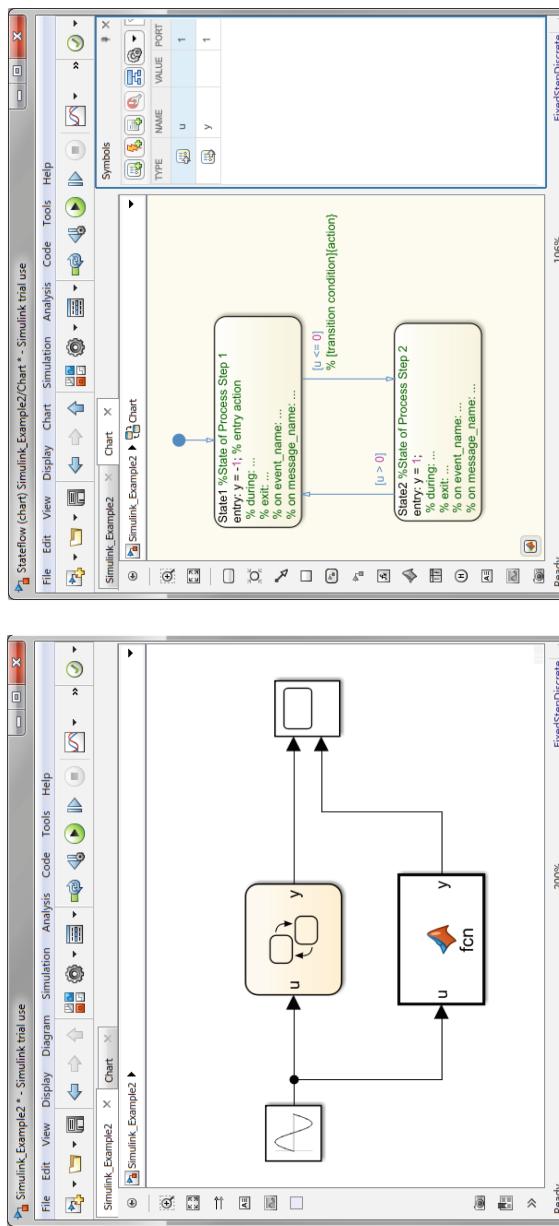
Figure 2.84 Example of MATLAB's ANN Construction (created to show the strength of the object of investigation)

provide standard plots, and support the data management efficiently. As Fig. 2.85 shows, even the integration of the *Deep Learning Toolbox* with the *Simulink* and *StateFlow* toolboxes can be visualized.

Example II: Fig. 2.85 (a) shows the visualization of a corresponding simulation design. A sine-wave signal is used twice: first, it is used to update states as they are specified in a stateflow block; and second, it is used as function input for a MATLAB code block. The output of both simulation blocks is finally visualized in a graphical plot. While the MATLAB code block considers an ANN built with the *Deep Learning Toolbox* to transform an input u to an output y , the transformation of the state flow block is specified in Fig. 2.85 (b). Any simulation input coming from the sine-wave is initially transformed to -1 , since a first state is activated. As this input refers to values greater than 0 (including zero), a second state is activated that transforms inputs to $+1$. During the simulation, even the flow between these two states can be animated by automatically highlighting the corresponding modeling shapes.

Critical appraisal: The strength of MATLAB clearly lies in the interplay of various toolboxes. On behalf of the toolbox called *Simulink*, complex simulations can be carried out efficiently. Graphical visualizations support an easy and failure-free simulation specification without the need of programming skills. The possibility of working on a codebase even allows users to specify complex simulation runs and greatly improves flexibility. Although 3D-based or AR-based animations are not considered at all, simulations can be animated on behalf of simple, graphical visualizations and an example can be found in the visualization of current states in state flow diagrams. During the simulation, each simulation step can be accessed by powerful control mechanisms. The robust date editor of MATLAB supports the generation and use of simulation data, so that data can be used for the training and testing of ANN and even considers various scenarios. So, in considering simulation modeling criteria, MATLAB satisfies technical perspectives.

Regrettably, modeling objects required from a process modeling and simulation perspective are not provided; process steps, as they are required in process models, can be circumstantially realized as states in state flow diagrams. Although the advantage of simulation-ready process models exists, which can efficiently be used in simulation runs on behalf of *Simulink*, the modeling process of processes and the interpretation of process models is distant from the intuitive and plausible dealing known from the process modeling carried out with process modeling tools.



On behalf of the toolbox called *Deep Learning Toolbox*, ANNs can be specified, trained, and tested efficiently and comfortably. Here, concrete ANN modeling shapes are provided in plenty. Even the use of ANN models in simulation runs is enabled, although the direct simulation of learning processes is very circumstantial and must be implemented manually. Further, although having numerous biological toolboxes, ANN models lack in biological realistic concepts and 3D models are not considered at all. So, modeling shapes rather focus on technical components such as different kinds of layers. The functioning, dimensioning, and positioning of individual neurons is, regrettably, not considered at all.

2.6.1.9 Critical Appraisal

The critical appraisal of process modeling and simulation tools is based on the following. First, approaches of a tool evaluation and selection are appraised. Then, process modeling and simulation tools are appraised. A process modeling and simulation tool-specific facet is then identified.

Tool Criteria: Most approaches to evaluate tools are based on a set of criteria. Since these are defined in order to identify attractive tools with regard to a certain evaluation purpose, which is not the construction or realization of a CoNM, an appropriate CoNM selection system, per se, is still missing. Hence, tools have neither been evaluated to be the foundation for the CoNM nor have they been evaluated to be a realization of the CoNM till date.

In order to decide whether current evaluation frameworks can at least partly identify a CoNM foundation or which criteria is suited to be part of a CoNM criteria catalog, the following appraises criteria category-wise:

1. Since the CoNM is intended to be carried out by commonly known tools, tool usage criteria focus on aspects which are accepted in the process modeling community. Criteria trying to apply accepted concepts from the process domain in the ANN process domain are highly relevant.
2. Since the CoNM is intended to be carried out by commonly known tools, technical concept criteria focus on technical aspects which are accepted in the process modeling community, simplify the implementation, and ensure the connection to practices and standards. Those are, therefore, highly relevant to be part of the CoNM criteria catalog as well.
3. As the CoNM is intended to be a first approach of Neuronal Modeling, user-specific criteria are only suitable for consecutive iterations because, from the perspective of the tool user, company-specific versions considering the company's size, industry, geographic positioning, etc., are instantiations of the gen-

- eral concept presented here. Hence, those are not required for the CoNM criteria catalog designed here.
4. From the perspective of the tool vendors, a CoNM is to be integrated in the company’s “ability to execute” and its “completeness of vision”. Those are modifications of the general concept presented here. Hence, those are not required for the CoNM criteria catalog designed here.

Since no single approach provided criteria about the use of neuronal techniques and concepts, which is an essential part of the ANN process domain, none of provided evaluation frameworks is able to even partly identify a CoNM foundation. This means that the process modeling domain has not yet been brought together with AI techniques. A research gap becomes visible here, which asks for the following: first, the construction of a CoNM because anything similar is missing; second, a CoNM tool selection system that extends common criteria and is able to evaluate current tools; and third, an analysis of currently available process modeling tools for the use of CoNM relevant aspects such as neuronal techniques, specific modeling objects, and common tool criteria.

Process Modeling Tools: Since modeling and simulation tools considered do not integrate with the ANN domain, corresponding modeling language criteria cannot be satisfied at all. As such, modeling and simulation tools only consider modeling objects inspired by traditional process and simulation research. While each supports the more or less interactive model construction, only some provide a graphical visualization and allow a modeling on behalf of modeling items.

First, there exist tools particularly supporting the construction of business processes models. Here, one can find tools such as the *ARIS Platform*, *ADONIS*, *Modelangelo*, and the *Signavio Process Editor*. While all provide a workspace for a graphical and interactive process model construction, and further provide a collection of standard analyses, all enable simple simulations referring to trivial KPIs, such as processing costs, probabilities, etc., with the exception of Modelangelo. Thus, more complex simulation runs are either hard or not at all realizable.

Second, there exist tools particularly focusing on the modeling of knowledge-intensive business processes. From the seven considered modeling and simulation tools, only the tool called *Modelangelo* enables the modeling of knowledge generation.

Third, there exist tools particularly supporting the construction of simulation models. Here, one can find tools such as *AnyLogic*, *SimPy* and *MATLAB*. Although SimPy provides a great flexibility in the creation of simulation models, models can only be specified on a codebase since a graphical modeling environment is not pro-

vided at all. This is only enabled by MATLAB and AnyLogic. Upon comparison, MATLAB dominates because of two reasons. First, it enables a more flexible construction of simulation models. Although AnyLogic allows the creation of arbitrary complex script files, the construction of simulation models with MATLAB on a graphical and code basis has blurred boundaries. Second, Matlab provides libraries for process simulations and deep learning algorithms. Regrettably, these two allow the more or less circumstantial use of ANNs in process simulations, but not their use in the sense of the CoNM, which refers to their integration with the ANN process domain.

Finally, there exist tools particularly focusing on the visualization of models constructed in three dimensions. Out of all modeling and simulation tools presented, one can only find *AnyLogic* and *Modelangelo* to apply to the above. While AnyLogic provides powerful modeling languages for the construction of simulation models, it provides a proper Space Markup Language (section 2.3.3.1) and its libraries provide 3D models so that simulations can be visualized in 3D animated runs. Regrettably, an augmentation within the real world is not considered at all. Although the tool called *Modelangelo* does not yet consider simulations, it is the only tool that, at the very least, provides prototypes of an AR-based process modeling and considers their simulation within the real world (Grum and Gronau, 2017; Grum and Gronau, 2018b).

Considering the four kinds of tools identified, no single approach is able to at least partly integrate with the ANN domain. Hence, ANN and Deep Learning tool criteria are not satisfied at all. Simulations focus on the realization of simple KPIs and probabilities, but not the biologically-plausible and neuromedically-correct model use. Machine learning algorithms and ANN model constructions available in MATLAB are rather limited and their use in the sense of a learning organization on behalf of neuronal techniques is not realized by any approach. A research gap becomes visible here, which asks for the following: first, the construction of a CoNM because anything similar is missing; second, a CoNM tool that satisfies common process modeling and simulation tool criteria and is able to consider the ANN domain as well as ANN and Deep Learning tool criteria; and third, an analysis of currently available process modeling and simulation tools for the use of CoNM relevant aspects such as process modeling and optimization concepts, specific modeling objects, and common modeling and simulation tool criteria.

Facit: Neither process modeling and simulation tool evaluation frameworks nor process modeling and simulation tools consider the integration of the process domain and the ANN domain. Criteria collections do not provide ANN and Deep Learning

criteria and tools do not realize the aforementioned criteria. Thus, a research gap within the process domain becomes visible here.

2.6.2 ANN Tools

Since terms are not used consistently by authors, different objects are referred to by the same term. This is even more complicated because of a shift of the understanding of the terms. As can be seen in the example of the term *framework*, over the last 10 years, authors speak about things having very little in common compared to each other (Stamer, Zimmermann, and Sandkuhl, 2016). Faced with a huge term heterogeneity, a common understanding from neuronal network tools is still missing. Some refer to *frameworks* and mention programming *libraries* (Jia et al., 2014), while others speak in general and mention *tools* or *computer programs*, their extension via *modules* or *packages*, and mention *environments* (Ye et al., 2016). Some even provide graphical user interfaces and integrate programming libraries and refer to *applications* (Gleeson, Steuber, and Silver, 2007). While authors refer to machine learning *systems* (Abadi et al., 2016), some mention *compilers* for mathematical expressions which stand beside non-ANN compilers, such as NumPy, Python, and SciPy, by which required algorithms can be implemented manually (Bergstra et al., 2010). Then, there exist *tools* which are dedicated to MEG/EEG/sEEG/ECoG data analysis from real brains (Tadel et al., 2011). Although those do not deal with ANN libraries, they deal with the processing and analysis of neuronal data, and concepts presented here are an inspiration for ANN techniques. Considering them all as ANN supporting tools, the following builds on the notational and most abstract term and defines the following:

Definition 31 (ANN Tools).

The ANN tool refers to an environment consisting of applications, frameworks, programming libraries, modules, and packages that support the specialized dealing with artificial neuronal networks and adapt from biological neuronal networks in accordance with the required repetitive, systematic, principal-based manner, which includes their development, training, testing, simulation, application, and research, such that model are constructed within a (subjective or) objective reality.

The term *specialized* in this definition stands for the neglect of simple tools, such as non-specialized compilers and non-specialized statistic tools, among others. This is because major ANN concepts are not provided here and implementation efforts were monumental.

Although the dealing with real neuronal networks will stand as fruitful inspiration for concepts from the domain of medicine, neuroscience, and common data analyses, the following focuses on the dealing with artificial neuronal networks because of the possibility of restricting learning to specific examples, to start learning from scratch, and to manipulate the learning progress itself. All this was neither possible with real, biological, neuronal networks (human brains) nor was it morally unscrupulous. Therefore, *Specialized data analysis* tools dedicated to data analysis from the real brain, such as *Brainstorm* (Tadel et al., 2011), EEG LAP, SPM, ERPLAB, and ERPWAVELAP, are put back in consecutive iterations of the CoNM.

The term *dealing* in this definition supports the application and the use of available ANN concepts and enables the research about neuronal networks in order to construct a CoNM. As only those kind of tools that enable *custom implementations* are considered for CoNM extensions, the following kinds of tools are neglected in addition. First, this includes tools enabling only customizations. Here, one can find *development environments* which allow one to modify and extend single components. Here, an implementation at will is not possible. Examples include *Peltarion Synapse*, *NeuroDimension*, *NeuroSolutions*, the *Scientific Software Neuro Laboratory*, and the *LIONsolver*. Free open source component based environments include *Encog* and *Neuroph*. Second, this includes tools enabling only simulations in order to research neuronal network structures and algorithms. Here, one can find *artificial NN simulators* such as the *Stuttgart Neural Network Simulator* (SNNS), *Emergent*, and *Neural Lab*. Further, one can find biological network simulators such as *Neuron*, *GENESIS*, *NEST*, and *Brian*. Third, this includes simulator tools for teaching neuronal network theory. Here, one can find tools only clarifying a collection of isolated concepts such as the *Parallel Distributed Processing Volumes* (PDI I-III), *PDP++*, *Emergent*, *tLearn*, *Basic Prop simulator*, and *Wintempla*.

Considering the definition presented here, the latter sections are structured as follows: the first sub-section presents common selection criteria; consecutive sub-sections provide selected ANN libraries and tools which will be considered as objects of investigation in section 4.2; and finally, tools and libraries are critically appraised in the last sub-section.

2.6.2.1 Selection of Adequate Tools

As the following will show, the selection of adequate ANN tools is mostly based on their performance with regard to a certain learning task. Hence, unlike process modeling tool evaluation approaches (see section 2.6.1.1), the selection is not only based on the fulfillment of certain criteria, but frameworks try to focus on their uniqueness. Further, market overviews on a regular basis are not available at all and criteria do not consider similar heterogeneous criteria categories as process modeling tool evaluation approaches do.

Criteria collections: The first category provides approaches that present, discuss, or interpret tools or libraries without comparing them with others. Here, the focus lies on their uniqueness and ideally, they are demonstrated on a small set of learning problems. Hence, those approaches are very suited for the identification of relevant tool criteria (or features) and the exemplar identification of tools fulfilling those criteria.

As the very first LSTM approach, *PyBrain* provides criteria focusing on the use of LSTM networks. Criteria can be grouped to categories such as the *supervised learning*, the *black-box optimization / evolutionary methods*, *reinforcement learning*, *architectures*, *compositionality*, *task and benchmarks*, and *speed optimization* (Schaul et al., 2010).

As a framework for scientists and practitioners, *Caffe* provides criteria for state-of-the-art deep learning algorithms (Jia et al., 2014). Criteria can be grouped into *software modularity*, *separation of representation and implementation*, *test coverage*, *Python and MATLAB bindings*, and the *provision of reference models*. In an only-fact-based comparison, the tools *Caffe*, *cuda-convnet*, *Decaf*, *OverFeat*, *Theano/Pylearn2*, and *Torch7* are subject to discussions.

As the most progressive approach, *neuroConstruct* provides criteria focusing on a 3D positioning of multicompartmental neurons. Criteria categories include the *importation and validation of morphologies*, *creation of simulator-independent conductance-based cell models*, *network generation*, *simulation management*, and *network analysis* (Gleeson, Steuber, and Silver, 2007).

Performance comparisons: The second category presents approaches that compare their performance with further tools. Here, the focus lies on the dominance of certain approaches rather than their functional uniqueness.

Focusing on large-scale machine learning, Abadi et al. (2016) present a performance comparison of *TensorFlow* with *Caffe*, *Neon*, and *Torch* for single-machine benchmarks and with *MXNet* for a synchronous replica microbenchmark. Besides the performance, criteria focus on the categories of *dataflow graph elements*, *partial and concurrent execution*, *distributed execution*, and *dynamic control flow*.

Providing a MATLAB-based framework for deep-learning, Ye et al. (2016) present a performance comparison of *LightNet* with *Caffe*, *Theano*, *Torch*, *TensorFlow*, and *Matconvnet*. Criteria focus on features and are grouped to the categories *architecture provision*, *demonstration provision*, *optimization algorithm provision*, *CPU and GPU support*, *efficient convolution computation*, and *hyper-parameter tuning automation*.

The library called *Theano* provides criteria for high-level description and efficient computation focusing on CPU vs GPU performance benchmarks (Bergstra et al.,

2010). Benchmarks consider the tools *Theano*, MATLAB with *GPUmat*, *Torch*, *EPLearn*, *NumPy*, *SciPi*, and *MATLAB*. Besides performance-related criteria such as *data entries per second*, criteria consider the *high-level description*, a *description and computation separation*, the *GPU and CPU computation*, the availability of *common machine learning algorithms*, and the *community*.

Identification of criteria: While some approaches only provide atomic criteria, others categorize criteria and present evaluations on an abstract level (Gleeson, Steuber, and Silver, 2007) and approaches become hard to compare. Because of approaches intersecting, the collection of approaches obtain redundant criteria. Hence, the following focuses on the identification of criteria independent from specific approaches, tries to establish a common criteria abstraction level, and appraises criteria individually such that a wide-ranging, redundancy-free perspective on tools can be constructed. As libraries do not only focus on NNs but consider further machine learning approaches, the following sets a focus on NN-relevant criteria only.

1) Tool Usage Criteria: The selection of adequate tools shall be based on the fulfillment of functional ANN demands (Gleeson, Steuber, and Silver, 2007), which support the use of tools.

1. **Usability:** The usability of tools can simplify the dealing with ANN tools and includes a variety, while the documentation and comprehensive installation clarify their use. Demonstration examples provide a warm start to new research and applications. This includes the availability of data for common learning problems such as the *Cart-Pole problem*, recipes, code examples, and even the availability of pre-trained reference models (Jia et al., 2014). Workflows, wizards, and routines, such as the automated hyper-parameter tuning, raise the comfort levels (Ye et al., 2016).
2. **Training Procedures:** Tools support common training methods (Ye et al., 2016). This sets a focus on supervised learning algorithms, classical gradient-based methods for sequential as well as non-sequential settings, etc., and considers evolutionary methods and methods of reinforcement learning such as Q-learning, SARSA, and REINFORCE (Schaul et al., 2010).
3. **Test Procedures:** Tools support common test methods. This sets a focus on supervised learning algorithms, considers evolutionary and reinforcement learning methods (Schaul et al., 2010), and includes the use of neural networks in simulations (Grum and Gronau, 2018a).

4. **Dataflow Graphs:** The tool considers *Dataflow Graphs* which are designed to represent all possible computations in a particular application. This includes primitive (functional operators, mutable states, operations that update it) and non-primitive operators (required for dynamic control flow), as well as data, which is transferred from operation to operation (Abadi et al., 2016). Besides the storage of models in configuration files, directed acyclic graphs realize the separation of representation and implementation so that instantiations can be issued independent from the construction (Jia et al., 2014). Here, alternatives refer to symbolic mathematical graphs (Bergstra et al., 2010).
5. **Creation of Cell Models:** Cellular mechanisms and cell models are to be defined in a simulator-independent format. This ensures, for instance, the consideration of unique conductance changes which occur because of the availability of specific voltage- and ligand-gated ion channels as well as the specification of complements and the density of these on the cell membrane (Gleeson, Steuber, and Silver, 2007). All of these are essential for the reproduction of complex behavior or real neurons. Since each cell model groups powerful mechanisms and is expected to be provided initially by the tool, specific models shall be considered as proper criteria and include simple cells modeled with transfer functions, such as the entity function, sigmoid function and logistic function, as well as axon hills modeled with aggregations, such as the summation and multiplication.
6. **Network Generation:** Tools must support the positioning of cell models within a region of 3D space and enable the generation of spatial, synaptic connections among them (Gleeson, Steuber, and Silver, 2007). This includes the layering of structures and the automated, guided, or rule-based synapse generation as well as the composing of custom neural network architectures (Schaul et al., 2010).
7. **Architecture Provision:** The tool must support common ANN architectures, such as feedforward architectures (time-independent), recurrent architectures (time-dependent), convolution networks, and LSTM blocks (Bergstra et al., 2010; Abadi et al., 2016; Ye et al., 2016). Since each architecture groups powerful mechanisms, architectures shall be considered independently.
8. **Compositionality:** Basic structures of the library enable the composition of different algorithms, compartments, architectures, etc. (Schaul et al., 2010). Therefore, LSTM architectures and recurrent architectures can stand side-by-side each requiring proper learning rules. Since those are not only connected but composed, they can be used and trained jointly as desired.
9. **Network Analysis:** The tool must support the realization of neuronal network analyses either internally or externally. This considers the various network

structures, learning behaviors, testing, and simulations, among others. In the library already including standard error plots, network connections plots and benchmarks as well as functionality for constructing, serializing, and deserializing data sets simplify the processing (Schaul et al., 2010). External neuronal simulations must consider commonly known, specialized simulation packages and are to be re-imported so that on-building analyses can be carried out (Gleeson, Steuber, and Silver, 2007). Here, common numerical analysis packages can help to realize consecutive analyses. State-of-the-art analysis concepts such as *Feature Visualization* group powerful mechanisms shall be considered independently (Zeiler and Fergus, 2013; Yosinski et al., 2015).

10. **Simulation and Analysis Visualization:** The tool must support the realization of neuronal network simulations from both internal and external simulation runs. External neuronal simulations must consider commonly known, specialized simulation packages and their results are to be re-imported so they can be visualized (Gleeson, Steuber, and Silver, 2007). Here, common numerical analysis packages can help to create visualizations.

2) Technical Concept Criteria: The selection of adequate tools shall be based on the fulfillment of functional ANN demands (Gleeson, Steuber, and Silver, 2007) which focus on the use of technical concepts.

1. **Augmented Reality Integration:** Tools must not only support the positioning of cell models within a region of 3D space and enable the generation of spatial, synaptic connections among them. They must also be considered within the real world so that their interactions with physical objects can be considered, size dimensions can be related to spatial attributes, and constitutions and physical phenomena can be applied realistically (Grum and Gronau, 2018a).
2. **Importation and Validation of Morphologies:** Morphologies and network architectures created in external programs and different formats are to be imported. This includes the reconstruction of models because of applications providing models with different abstraction levels, variations in compartment numbers, or the use of different conductance models (Gleeson, Steuber, and Silver, 2007). As errors are checked continuously and can be detected and managed even during the importation process.
3. **Simulation Management:** Common simulator environments such as NEURON or GENESIS are to be integrated with the tool. This can be carried out by the automated translation of internal models to simulation environment specific script files. Simulation results being executed by those script files can then be re-imported since they are stored in standardized text files (Gleeson, Steuber, and

Silver, 2007). As various scenarios are to be simulated, identifiers separate simulation results which all demand for a simulation management. The realization of simulation variations is particularly challenging in regard to bookkeeping (Greff et al., 2017). Hence, tools must support a simulation management.

4. **Data Management:** Since the dealing with neuronal networks can be connected to huge data sets coming from various technical systems, such as live systems, analytics environments, production systems, off-line data sources, and data streams, their integration with ANN tools can become very complex (Dean and Ghemawat, 2008; Armbrust et al., 2015; Moritz et al., 2015). Particularly, the realization of experiment run variations considering different training methods, data structures, data types, data sources, and machine learning libraries pose challenges in bookkeeping (Greff et al., 2017). Hence, tools must support data management.
5. **Speed Optimization:** The benefit of short development cycles and small lines of code with high-level programming languages is at the expense of the speed, especially during training, test, and simulation runs, of low-level programming languages and vice versa (Ye et al., 2016). *Bridging components and parallel implementations*, therefore, can help to mitigate this problem (Schaul et al., 2010). *Bindings* to common libraries expose easy prototyping and interfacing with existing research code (Jia et al., 2014). Further, this includes the use of efficient algorithms in multicore computing processing units (CPU) and the use of fast graphical processing units (GPU) (Bergstra et al., 2010; Jia et al., 2014; Abadi et al., 2016; Ye et al., 2016). Ideally, algorithms for the efficient computation are separated from the *high-level descriptions* and automated routines translate high-level descriptions to computation algorithms (Bergstra et al., 2010).
6. **Multi-Platform Support:** Tools must support the use of various platforms, specific hardware characteristics, and the current hardware need of neuronal networks. During training and testing, hardware-intensive operations can be carried out at large distributed clusters in a data centers and the use of neuronal networks in production can be running locally on mobile devices (Grum et al., 2018). Here, a *parameter server architecture* supports the specification of hardware characteristics and the management of states and processes for partial, concurrent, or sequential operations. Further, the *deferred execution method* is suitable: the first phase defines the program and the second phase executes an optimized version of the program on the set of available devices (Abadi et al., 2016).

3) Vendor specific criteria: Vendor-specific criteria have to be considered similar to the domain of process tools.

1. **License:** Being part of a research community and following the intention to contribute with the CoNM possibly in open source projects so that it can be freely used, modified, and shared, tools must be provided by a license being approved by the *Open Source Initiative* (OSI) such as the Apache license (Abadi et al., 2016) and BSD license (Schaul et al., 2010; Jia et al., 2014) as only some works are.
2. **Community:** The tool is supported by a community. As problems are identified, questions occur, and ideas arise, the community can help to make personal issues subject to discussion. This includes the creation of standards such as automated test procedures that are demanded for any module before it is accepted in the community (Jia et al., 2014). The management of communities is recommended to be connected to custom-oriented mailing lists, such as *announce* mailing lists, *developer discussions*, and *user support* (Bergstra et al., 2010).

Interim Conclusion about Evaluation Approaches: Beside the option of a new development, any tool from the ANN domain can be attractive for the critical appraisal about its suitability to be extended with CoNM concepts presented here. This includes *batch dataflow systems*, which obtain variations of powerful *MapReduce* approaches (Dean and Ghemawat, 2008) such as *SparkNet* (Moritz et al., 2015). Because of their limitation to require input data to be immutable and all of the subcomputations to be deterministic (only then the system can re-execute subcomputations when machines in the cluster fail), updating a machine learning model is an expensive operation (Abadi et al., 2016, p. 268) and slows the convergence of training processes (Moritz et al., 2015). Hence, these are not considered for the CoNM at all.

As the performance of tools with regard to a certain domain is not relevant for CoNM, since the CoNM intends to be generalizable, performances of libraries or tools are not considered for the selection as well. Since tools are selected by a best overall criteria fulfillment, best candidates are especially suitable to be part of this CoNM evaluation.

Further, simple compiler and non-specialized tools such as *NumPy*, *MATLAB*, and *SciPi* have been excluded. Although they can be extended, they are not suited to be extended because of the immense implementation efforts required.

Faced with a huge number of ANN tools and libraries mentioned in previous selection system approaches, the following intends to establish a reasoning for a

preselection of tools to be evaluated in a first step such that a suitable foundation for the CoNM can be identified in a second step:

1. Tools having a broad market distribution.
2. Tools having a broad acceptance in the ANN community.
3. Tools being applied in at least one scientific publication.

2.6.2.2 PyBrain

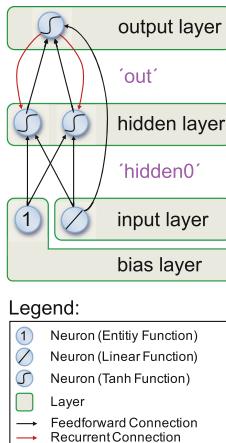
PyBrain is a programming library for *Python* that offers easy-to-use algorithms for machine learning tasks (Schaul et al., 2010). It provides algorithms for neural networks for reinforcement learning (including their combination) and for unsupervised learning and evolution, which can be extended easily. This is the reason for its name, which stands for **P**ython-Based Reinforcement Learning, Artificial Intelligence and Neural Network Library (Bayer et al., 2018).

Description: Although it is intended to be used by entry-level researchers, it is flexible for use in state-of-the-art research (Schmidhuber, 2013). This includes the management of data required for machine learning tasks, the model construction on behalf of predefined trainers, and the realization of classification and validation tasks during the learning run (IDSIA, 2009). As the data preprocessing, the result use, and the visualization are not part of PyBrain itself, these tasks can be realized by manual implementations on behalf of Python.

Side facts: Originally, it was developed by the team of Prof. Jürgen Schmidhuber at the Dalle Molle Institute for Artificial Intelligence in Switzerland and the Technical University of Munich in Germany in 2010. Licensed under the *BSD Software License*, it is now developed by the PyBrain community and is available under its entire repository (IDSIA, 2018).

Example: The simplicity of its use can be seen in Fig. 2.86. It shows a code example for solving the parity problem with a recurrent neuronal network on the right and the network's visualization on the left.

At the listing, one can see the simple data set initialization with one line of code (line #2). The building of an ANN is automated and needs only four lines of code (lines #5–#8). The training demands for two further lines of code (lines #11–#12). Although the dealing with ANN normally is much more complex, since for the first, data sets are not prepared perfectly in advance, for the second, network structures follow individual designs, and for the third, advanced plots are required, the example shows an efficient dealing with ANN only on behalf of a given set



```

1 # Load Data Set.
2 ds = SequentialDataSet.loadFromFile('xor.mat')
3
4 # Build a recurrent Network.
5 net = buildNetwork(2, 2, 1, bias=True, hiddenclass=TanhLayer,
6                     outclass=TanhLayer, recurrent=True)
7 recCon = FullConnection(net['out'], net['hidden0'])
8 net.addRecurrentConnection(recCon)
9 net.sortModules()
10
11 # Create a trainer for backprop and train the net.
12 trainer = BackpropTrainer(net, ds, learningrate=0.05)
13 trainer.trainEpochs(1000)

```

LISTING 2.5: PyBrain Code Example.

Figure 2.86 Code Example for Solving the Parity Problem with PyBrain (created to show the strength of the object of investigation)

of functions. With these, results can be accessed as simple console printouts and a quick start is supported. So, an entry for beginners is simplified. Individual network structures, challenging data preprocessing activities, and visualization issues can then be realized by non-beginners on behalf of individual implementations.

Critical appraisal: Since PyBrain does not offer any interface for modeling or simulation tools (API), no modeling language criteria can be satisfied. The business process context is not considered at all which is reasonable when faced with the particular focus of PyBrain as a programming library to algorithmically deal with artificial neuronal networks. Hence, modeling shapes that go beyond algorithmically dealing in the sense of the machine learning can hardly be extended with extensive implementation efforts. This includes the integration with simulation environments. It can only be realized with high implementation efforts because PyBrain was not initially intended to be used in simulation environments. Here, only basic mechanisms such as the activation of the entire ANN are provided by PyBrain. Since at least an algorithmic foundation is available, modeling shapes for biological models of neuronal networks and modeling entities for artificial neuronal networks can be supplemented with adequate implementation efforts.

PyBrain's focus clearly lies on the dealing with powerful LSTM networks. Hence, common ANN tool criteria can be satisfied. The only exception here are Convolutional network architectures and speed-optimized training procedures which are considered only in later versions of its Git repository (IDSIA, 2018). Issues about feature visualization, management of training procedures, and realistic 3D modeling of neuronal compartments are not considered at all.

2.6.2.3 Theano

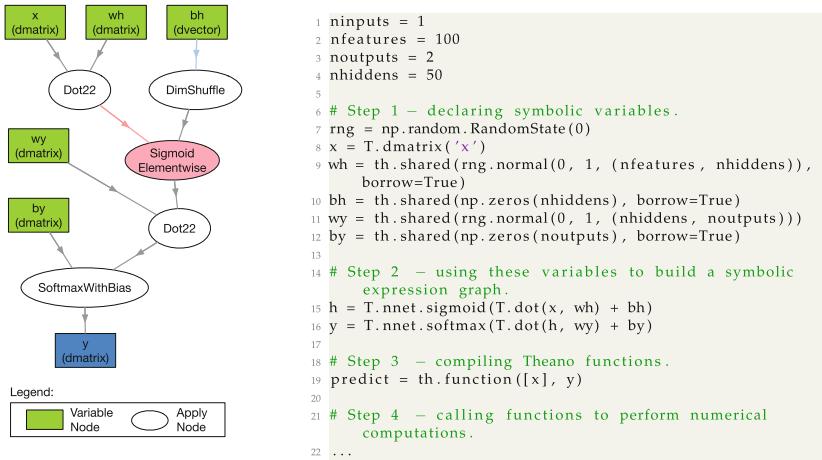
Theano is a programming library for *Python* that offers easy-to-use algorithms for efficiently defining, optimizing, and evaluating mathematical expressions on behalf of multi-dimensional arrays (The Theano Development Team et al., 2016).

Description: Since mathematical expressions implemented and interpreted by Theano allow a symbolic differentiation before their performing, Theano automatically optimizes the choice of expressions and translates them into *C++* or *CUDA* code (Bergstra et al., 2010). So, an efficient realization on behalf of either the CPU or GPU can be realized.

Side facts: Initially licensed under the *BSD Software License*, it was written at the LISA lab of the Montreal Institute for Learning Algorithms (MILA) at the Université de Montréal (LISA lab and Team, 2018b). Since 2008, it has been actively and continuously developed by its community (LISA lab and Team, 2018a). Theano is named after the Greek mathematician, who may have been Pythagoras' wife.

Example: The simplicity of its use can be seen in Fig. 2.87. It shows a code example for solving the parity problem with a MLP on the right and the Theano-generated network visualization on the left.

At the listing, one can see the declaration of variables needing six lines of code (lines #7–#12). Its design is highly dependent on the network architecture. Although this seems laborious, this is an effective approach for large and regular architectures. If special architectures need to be defined, for instance, because of anomalies intended or the intention to manage the meaning of substructures, this process is a challenging and error-prone one. The building of an ANN is finalized in a second step. Here, symbolic expressions are defined on behalf of variables defined (lines #15–#16). The third step compiles Theano functions for an optimized performing (line #19), before functions can be used in a fourth step. The training and data preparation demands for a manual treatment (line #22). Since this demands for a great number of lines of code, the corresponding lines have been conserved.



LISTING 2.6: Theano Code Example.

Figure 2.87 Code Example for Solving the Parity Problem with Theano (created to show the strength of the object of investigation)

Critical appraisal: Theano includes powerful tools for manipulating and optimizing graphs representing symbolic mathematical expressions (Bastien et al., 2012). This allows users to quickly prototype complex machine learning models, to quickly deal with gradient descent methods without manually deriving the gradient, to decrease the amount of code required, and to eliminate practitioner errors. Beside the creation of static printouts and pictures on behalf of *theano.printing.pydotprint*, an interactive HTML version can be created on behalf of the *d3viz* module (The Theano Development Team et al., 2016). Here, users can zoom into different regions, move graphs via drag and drop, and position nodes both manually and automatically. Further, additional information about nodes and edges can be visualized and saved, such as their data type or definition in the source code, and the expansion or shrinking of nested graphs such as *OpFromGraph* nodes can be accessed. Node labels and profiling information of nodes even can be edited here and graphical analyses, such as the time required for the performing at a certain layer, can be mapped onto defined layers. An example for the interactive expansion of an *OpFromGraph* node can be seen in Fig. 2.88.

Although those graphs represent dataflow and function application within the network architecture, the individual neuron, over the whole layer, is not considered

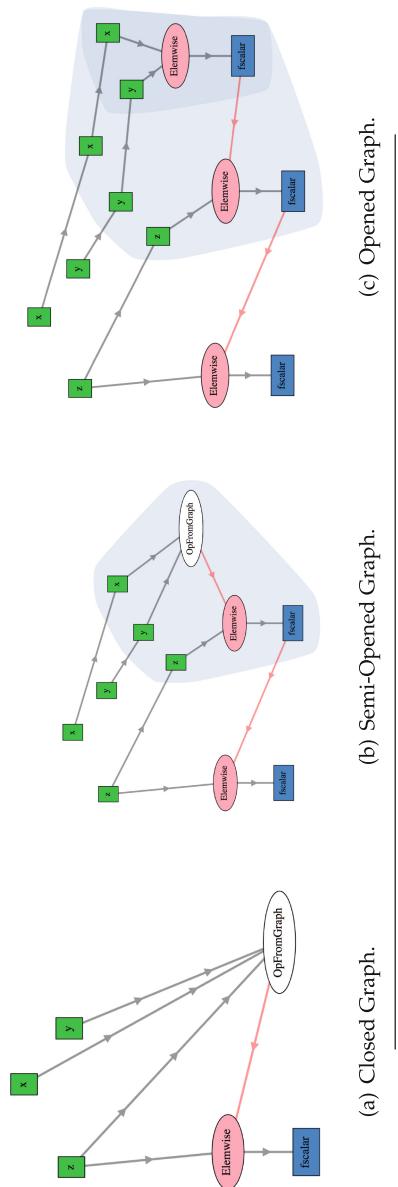


Figure 2.88 OpFromGraph Node Example of Theano (created to show the strength of the object of investigation)

at all. Further, insights made because of the network activation, for instance, in simulations, are not visualized at all. Theano's visualizations only consider the static network structure and the efficient processing at different machines. Here, one has to distinguish between the efficient performing of neuronal networks at different machines and the efficient structure of the network itself. While the first demands for the selection of adequate machines, the latter is relevant for the identification of a data object's context, its processing position within the network, and the subsequent generation of knowledge at a specific neuron. So, graphs can stand as a foundation for the expansion with knowledge modeling criteria and simulation criteria. Although an interaction with graphs is possible, this is limited to the editing of only some variables and the examination of network structures. Thus, the graphical design of network architectures cannot be realized. Modeling and simulation tool criteria, therefore, cannot be met at all.

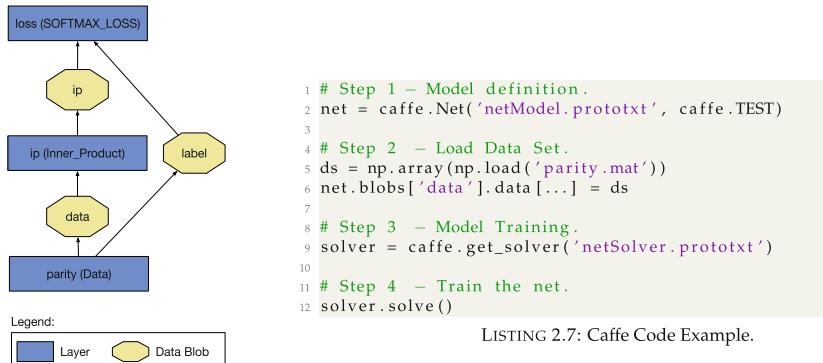
The implementation effort required for modeling language criteria with regard to biological neuronal networks and artificial neuronal networks is adequate since basic algorithms of a machine learning are available with Theano. Aspects of the domain of business processes and simulation are not available at all. Hence, only ANN tool criteria can be satisfied and demand for adequate implementation efforts.

2.6.2.4 Caffe(2)

Caffe is a programming library for *C++* that offers a modifiable framework for state-of-the-art deep learning algorithms and a collection of reference models (Jia et al., 2014). Its focus lies on fast and performance-effective dealing with convolutional architectures.

Description: The performance-effective dealing with artificial neuronal networks is based on the separation of network representation and implementation. Model definitions, data origins, and training settings are stored in configuration files (using the Protocol Buffer Language), so that their modification and exchange and their distributed development and implementation can be realized easily. Since Caffe can be embedded on CUDA hardware, mobile, and desktop devices, the switching between the use of CPU, GPU, and platforms supports an efficient processing.

Side facts: Although it is intended to be used by multimedia scientist and practitioners, its flexible use can enable state-of-the-art research. It is licensed under the *BSD Software License* and provides bindings for *Python* and *MATLAB*. Originally, Caffe was developed by the Berkley AI Research Laboratory (BAIR, 2018d) and its community contributors (BAIR, 2018c). Now, it is continued as Caffe2 under the support of Facebook and Google (BAIR, 2018b) and its community (BAIR, 2018a).



LISTING 2.7: Caffe Code Example.

Figure 2.89 Code Example for Solving the Parity Problem with Caffe (created to show the strength of the object of investigation)

Example: The simplicity of its use can be seen in Fig. 2.89. It shows a code example on behalf of Python for solving the parity problem with a recurrent neuronal network on the right and the Caffe-generated network visualization on the left.

In the listing, one can see that the building of the ANN is realized in two steps: first, the ANN architecture needs to be defined and stored in a *model configuration file* with the extension .prototxt; and second, the model configuration file is loaded (line #2). An example for the model of the parity problem can be found in Fig. 2.90. A great community collection of models enriches this process. Then, the data set is loaded (line #5) and related to the network (line #6). A conversion of the data to an appropriate format, the *leveldb* or *lmdb* format, must be realized manually. The training is realized in two steps. First, the training needs to be specified and stored in a *solver configuration file* with the extension .prototxt. An example for the solver of the parity problem can be found in Fig. 2.90. Second, the solver configuration file is loaded (line #9). The training is realized with one more line of code (line #12). Of course, more sophisticated function provide more control. Alternatively, it can be started from the console. Since trained models are stored in a file with the extension .caffemodel, they can be easily accessed. Although the separation of definitions and implementations demand for more line of codes, the advantages for this clearly lies on the raised flexibility.

```

1 name: "ParityExampleNet"
2 layer {
3   name: "parity"
4   type: "Data"
5   top: "data"
6   top: "label"
7   data_param {
8     source: "input_leveldb"
9     batch_size: 4
10  }
11 }
12 layer {
13   name: "ip"
14   type: "InnerProduct"
15   bottom: "data"
16   top: "ip"
17   inner_product_param {
18     num_output: 2
19   }
20   bias_filler {
21     type: "constant"
22   }
23 }
24 layer {
25   name: "loss"
26   type: "EuclideanLoss"
27   bottom: "ip"
28   bottom: "label"
29   top: "loss"
30 }
```

LISTING 2.8:
Caffe Code Example Model
'netModel.prototxt'.

```

1 net: "./netModel.prototxt"
2 test_iter: 1000
3 test_interval: 1000
4 base_lr: 0.001
5 lr_policy: "step"
6 gamma: 0.1
7 stepsize: 2500
8 display: 50
9 max_iter: 40000
10 momentum: 0.9
11 weight_decay: 0.0005
12 snapshot: 5000
13 snapshot_prefix: "./netModel"
14 solver_mode: GPU
```

LISTING 2.9: Caffe Code Example Solver
'netSolver.prototxt'.

Figure 2.90 Configuration Files for solving the Parity Problem with Caffe (created to show the strength of the object of investigation)

Critical appraisal: Based on the configuration files defined, Caffe can visualize network architectures in the form of arbitrarily-directed acyclic graphs. An example can be found in Fig. 2.89). These even visualize the dataflow between layers. Regrettably, those visualizations are only accessible by PNG files. Hence, the intuitive, bidirectional modification of architectures, much less a modeling, is not possible at all. Further, models of the dataflow only consider the most abstract flow of all the data from its origin through the network to its sinks. So, the dataflow is decoupled from individual network compartments, its instantiated, and time-dependent changes as well as the meaning within the current network context. Hence, a high implementation effort is required for modeling language criteria of the knowledge and perspective domain.

Since the business process and simulation context is not focused on by Caffe, the corresponding modeling elements and tool criteria cannot be satisfied; they can hardly be extended on behalf of circumstantial implementation efforts.

Caffe's focus clearly lies in the dealing with powerful and efficient Convolutional networks. It even comes with a collection of CNN models such as *AlexNet* and *GoogLeNet*. Hence, common ANN tool criteria can be satisfied, but a management of training procedures and a realistic 3D modeling of neuronal compartments demand manageable implementation efforts. Since basic mechanisms for an algorithmic foundation are available, modeling shapes for biological neuron models and modeling entities for artificial neuronal networks can be supplemented with an adequate implementation effort.

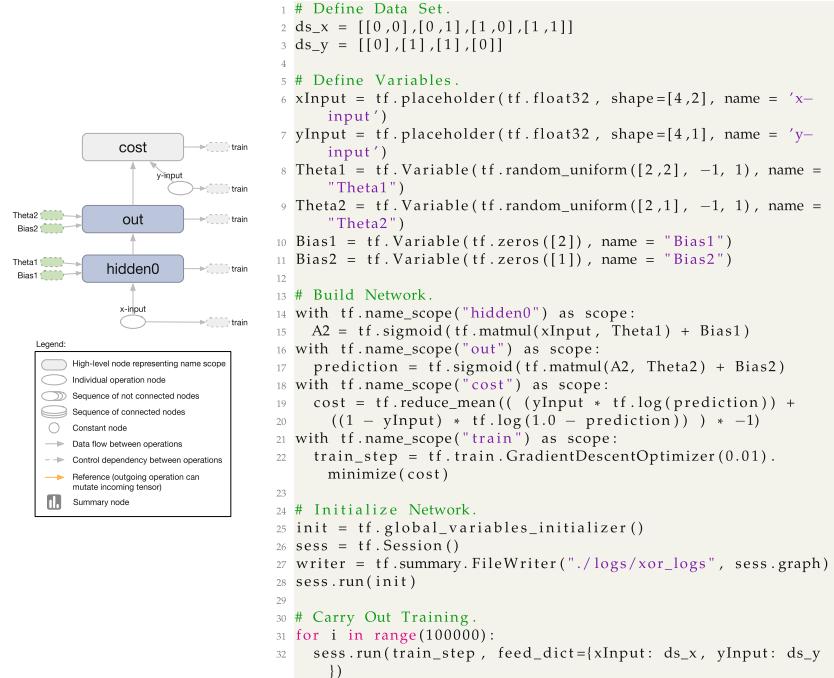
2.6.2.5 TensorFlow

TensorFlow is a framework for data flow programming, which includes a programming library implemented in *C++* and *Python* as well as high-level APIs and tools for the dealing with machine learning projects (Abadi et al., 2016). It is named after the algorithmic dealing with *tensors*, the general term for a unit of data here.

Description: Each tensor denotes a set of primitive values shaped into an array of any number of dimensions. A set of tools supports the creation of computational graphs on the basis of the occurrence of tensors, which show the series of TensorFlow operations arranged into a graph of nodes. It is accessed by a component called *TensorBoard* that visually presents the structure of a graph in a browser. Due to its flexible architecture, the efficient performance of numerical computations can be realized on a variety of platforms. Here, one can find CPUs, GPUs, and TPUs of desktop, cluster, server, mobile, and edge machines (Abadi et al., 2015).

Side facts: Although it intends to provide state-of-the-art machine learning algorithms, its flexible numerical computation core is used across many other scientific domains (Google-Brain-Team, 2018). Originally, it was developed for internal use at the Google's AI laboratory. Now, it is licensed under the *Apache 2.0* Open Source software license and it is developed by its community (Google-Brain-Team and Community, 2018b).

Deployed under the *Apache 2.0* Open Source software license, TensorFlow can be supplemented by *LUCID*, which is a state-of-the-art collection of tools for research regarding neuronal network interpretability (Google-Brain-Team and Community, 2018a). This empowers TensorFlow with issues particularly about feature visualization, attribution, and parameterization (Han et al., 2018; Bhardwaj, Suda, and Marculescu, 2019; Furnell et al., 2019; Ray et al., 2019). Since LUCID must be seen as individual programming library, the combination of TensorFlow and LUCID are considered as separate OoI.



LISTING 2.10: TensorFlow Code Example.

Figure 2.91 Code Example for Solving the Parity Problem with TensorFlow (created to show the strength of the object of investigation)

Example: An example visualization can be seen in Fig. 2.91 on the left. The set of modeling shapes can be found within the legend. Its corresponding code is shown on the right and represents an example for solving the XOR problem with a simple feedforward neuronal network.

At the listing, one can see the definition of the data set with two lines of code (lines #2–#3). Variables required are defined from lines #9–#14. Then, the network is built by lines #17–#25. For this, each layer can be grouped by an individual name scope. After its initialization (lines #28–#31), the training can be carried out with lines #34–#35. While the TensorBoard can visualize graphs enriched by metadata of runs, such as total memory or total compute time, it enables an interactive visualization design. Interaction possibilities include the expansion and shrinkage

of subnetworks, the exclusion of unnecessary nodes and edges by shortcuts, the identification of similar network parts by the same color, or the visualization of the edge thickness, which reflects the total tensor size (Wongsuphasawat et al., 2018). Regrettably, an interactive network design or the positioning of network parts cannot be carried out with help of the TensorBoard. So, network architectures have to be designed on a codebase and the position of its compartments is only generated algorithmically, which does not create biologically-plausible graphs.

Critical appraisal: In general, TensorFlow provides a set of modeling objects for the visualization of the data flow, only missing the modeling item for a data object. Since TensorFlow's focus clearly lies on the efficient computation, operations of neuronal compartments and operations of a non-neuronal compartments are visualized side by side. Examples can be found in initialization operations, training operations, modification operations, etc., that stand apart from high-level nodes representing network compartments. Hence, a focus on neuronal compartments including the neuron-specific creation of knowledge, information, and data objects cannot be established. Further, modeling language criteria of the process and simulation domain are not considered at all.

TensorFlow clearly places a focus on machine learning and ANN. Particular visualizations provided by *TensorBoard* are very helpful for the course of training and hyper-parameterization. Over the horizon of ANN tools, modeling and simulation tool criteria cannot be satisfied. Since LUCID enables TensorFlow with powerful approaches of a feature visualization and attribution and builds on TensorFlow, it supplements ANN modeling and ANN tool criteria presented here.

2.6.2.6 NeuroConstruct

NeuroConstruct is an ANN tool implemented in *Java* for the graphical model design, visualization, and analysis of realistic network models considering multicompartmental neurons in 3D space (Gleeson, Steuber, and Silver, 2007).

Description: The dealing with graphical models includes models incorporating dendritic morphologies and realistic cell membrane conductances, such as inhibitory and excitatory synaptic input (Rothman et al., 2009). Its graphical user interface allows model creation and modification without programming. Since models are expressed on the basis of a simulator-independent standard, which is further on the basis of the XML format called *NeuroML standard* (Gleeson et al., 2010), the automated code generation enables the model exchange with neuronal simulators such as *NEURON* (see section 2.6.2.8) and *GENESIS* (see section 2.6.2.9).

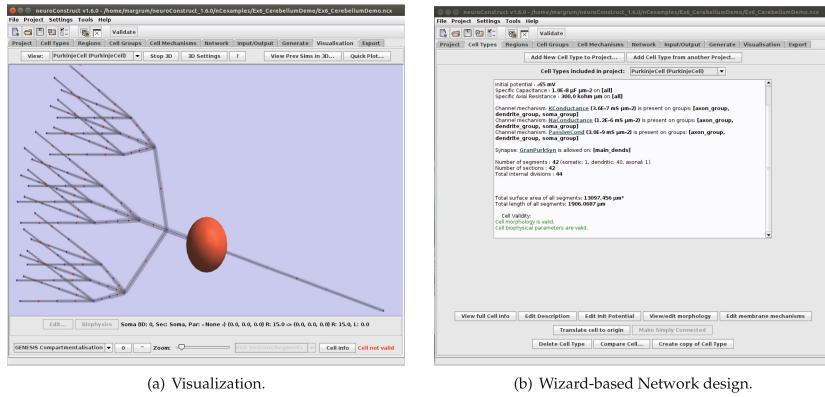


Figure 2.92 Example of neuroConstruct (chosen to show the strength of the object of investigation, De Schutter and Bower, 1994)

Side facts: Originally, it was developed by the *Silver Lab* in the Department of Neuroscience, Physiology and Pharmacology at UCL London’s Global University (neuroConstruct-Team et al., 2018b). Now, it is developed by its community (neuroConstruct-Team et al., 2018a) under the *GNU General Public License v2.0* Open Source software license.

Example: The visualization power of its use can be seen in Fig. 2.92 that shows the network definition on behalf of wizards. On the right, one can see the definition of the *Purkinje Cell* (De Schutter and Bower, 1994), which is part of neuroConstruct’s example repository. On the left, one can see the tool-generated network visualization.

Wizards required for the modeling of neuronal networks in 3D are visualized tabulator-wise in the tool. The first wizard manages meta-information required for the management of the project. The second wizard, which defines *cell types*, can be seen to the right of the picture. Then, *regions* and *cell groups* are defined (third and fourth tabulator). Individual cell mechanisms are defined in a fifth tabulator. The network connections, including their inputs and outputs, are defined thereafter (tabulator six and seven). Before the 3D network visualization can be generated in the ninth tabulator, the positioning needs to be generated by neuroConstruct in the eight tabulator. Network structures can be exported for the external use in the last tabulator. Since wizards systematically guide through the creation process and ensure adequate syntax and rule use, visualizations can be generated on a failure-tolerant basis. As those can be translated into various programming languages for

external simulation tools, simulation results can even be efficiently imported and considered in 3D visualizations. The analysis figure creation (*histograms* of spike times, numbers of synaptic connections or interspike intervals, *raster plots*, *voltage traces*, and *correlation plots*) clearly supports the analysis (Gleeson, Steuber, and Silver, 2007).

Critical appraisal: Since neuroConstruct focuses on the creation of 3D plausible models of neuronal networks, it does not integrate with the process domain at all. Therefore, no modeling item is provided in the sense of modeling shapes and compartments only refer to the physical space required, including their biological mechanisms. NeuroConstruct does not even fulfill requirements of modeling or simulation tools because mechanisms and models are not carried out in proper simulations.

With exceptions, ANN tool criteria are satisfied. First, on behalf of groups and layers, arbitrary complex hierarchies of neuronal network structures can be generated. This realizes powerful overviews and a management of hierarchies. Regrettably, an interactive shrinkage and expansion of compartments cannot be realized. Isolated views of compartments provide an individual focus, but they do not enable the visualization of individual parts in interplay with the entire network. Further, the denomination of the processual meaning of compartments is not possible at all.

Second, although algorithms such as the *morphology-based connection algorithm* or the *volume-based connection algorithm* simplify the creation of large network structures (Gleeson, Steuber, and Silver, 2007), the generation only includes the automated positioning and connection on the basis of algorithms. An arbitrary positioning and connecting in the sense of an interactive modeling cannot be realized at all. So, visualizations only enable an insight but not a creative design process comparable to the process modeling.

Since morphologies can be imported and cellular mechanisms are able to be specified individually, neuronal simulations can be plausibly realized. This process disregards questions regarding how a functioning structure can be identified. Commonly, structures can follow observations of real cells, but the artificial creation on behalf of machine learning algorithms is not considered at all.

2.6.2.7 NeuroLOTs and NeuroTessMesh

NeuroTessMesh is a tool for the reconstruction and visualization of 3D meshes from neuronal compartments. It is part of the set of libraries and tools called NeuroLOTs and functions as front-end GUI to the NeuroLOTs framework.

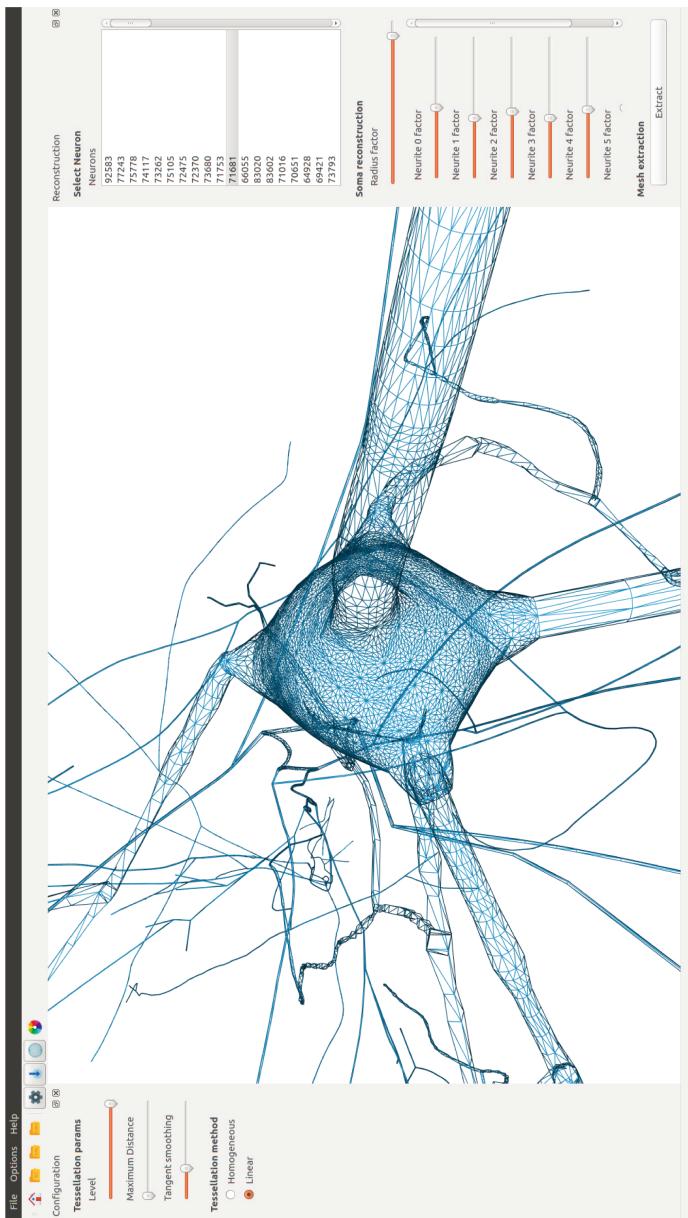


Figure 2.93 Example of NeuroTessMesh (GMRV, 2018)

Description: Mesh forms are generated from morphological tracing information. Providing any freedom to rotate and move in 3D space, even a selection of parameters can be edited manually (e.g. soma radius, neurites' starting points). Further, tessellation operations can be dynamically realized by GPU operations so that an efficient modification and good quality visualizations of complex neuronal scenes can be realized (Garcia-Cantero et al., 2017).

Side facts: NeuroTessMesh is named after the 3D object modeling with mesh structures, what is called mesh, and the mathematical tiling of a surface using one or more geometric shapes, the so called tiles. Originally, it was developed by the GMRV -the Virtual Reality and Modeling Group of the University Rey Juan Carlos (GMRV, 2018). Now, it is developed by its community in separate repositories (GMRV and Community, 2018a as well as GMRV and Community, 2018b).

Example: An example, which gives an impression of the 3D modeling on behalf of NeuroTessMesh can be found in Fig. 2.93.

Here, one can see the selection of a neuron called 71681. The parameters required for its tessellation can be modified on its left and right. Although an interaction is enabled by the modification of those parameters, an intuitive modeling per drag and drop, as it is known from the process modeling domain, cannot be realized yet.

Critical appraisal: Hardly any modeling language criteria aspect can be satisfied by NeuroTessMesh since it does not integrate with the process domain. Further, its focus only lies on the 3D visualization of neuronal networks. So, meshed models are not considered in a machine learning sense and ANN and DL tool criteria are not satisfied either. Meshed models only serve for the preparation of simulations and thus, model ultrarealistic and efficient 3D information.

2.6.2.8 NEURON

NEURON is an environment for the biologically-realistic simulation of signaling procedures in neurons and neuronal networks. This includes both electrical and chemical signaling (Hines and Carnevale, 1997). It provides tools for conveniently building, managing, and using neuronal models in simulation runs. Adequate simulation control and graphical presentation mechanisms allow an intuitive modeling and modification on behalf of parameters and thus, enable direct dealing with neuroscience concepts.

Description: The NEURON environment allows the modelers to address high-level research questions without thinking about the low-level mathematical func-

tioning and computational issues. For example, spatial and temporal discretization issues are automatically considered by NEURON (Carnevale and Hines, 2006). Further, a C-like, extension of *hoc* has been built as an interpreted programming language (Kernighan and Pike, 1984). It allows the dealing with functions specific to the domain of modeling neurons, the dealing as object-oriented programming, and the modeling by wizards and graphical interfaces. So, models can be created efficiently on a codebase. Then, they can be visualized by graphical tools and used in neuroscience simulations.

Side facts: It has a research and development history of over thirty years, starting in 1976 in the laboratory of John W. Moore at Duke University, and has been widely used in research in more than 1900 scientific articles and books (Moore's laboratory and Team, 2018). Now, it is developed by its community (NEURON-Community, 2018) under a three-clause *BSD* Open Source software license.

Example: An example according to Carnevale and Hines (2006), which gives an impression of the modeling of a nerve cell on behalf of NEURON's *hoc*, can be found in Listing 2.11.

According to Hines and Carnevale (1997), the example requires four steps: first, the model topology is established; second, anatomical and biophysical properties are assigned; third, stimulating electrodes are attached; and fourth, the simulation time course is controlled. As the listing shows, a modeling on an abstract level is scripted and corresponding mechanisms for the chemical and electrical signaling are carried out in the background.

Beside the artificial neuronal model and simulation model specification on behalf of the scripting language *hoc*, the tool provides the possibility to specify and control corresponding models on behalf of wizards and GUI-based mechanisms. As can be seen in Fig. 2.94, the network can be generated in a drag-and-drop manner, it can be controlled by clicks on buttons, and results are visualized by different kinds of plots.

In this figure, NEURON's main menu can be seen in the top-left window. Individual parameters can be specified in the center-left window. The topology of the network is prepared in the bottom-left window and simulation results are visualized on behalf of two graph windows on the right. The simulation is controlled by the center-top window called *RunControl*. Individual time steps can be controlled by the bottom-center window called *MovieRun* such that the spike trains generated by the simulation can be visualized like a movie.

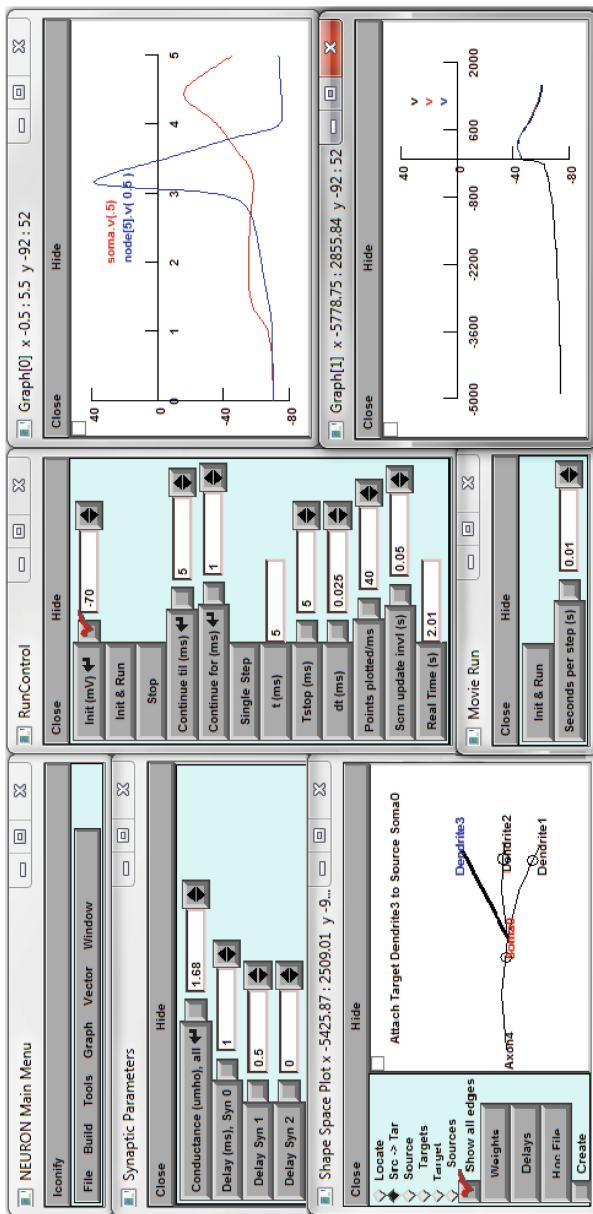


Figure 2.94 Code Example for a Soma Simulation using NEURON (created to show the strength of the object of investigation)

```

1 # Step 1 - model topology.
2 create soma, axon, dendrite[3] // declare sections
3 connect axon(0), soma(0)           // attach sections to each
4   other
5   for i=0,2 {
6     connect dendrite[i](0), soma(1)
7   }
8
9 # Step 2 - specify anatomical and biophysical properties.
10 soma {
11   nseg = 1 // compartmentalization parameter
12   L = 50 // [mm] length
13   diam = 50 // [mm] diameter
14   insert hh // standard Hodgkin-Huxley currents
15   gnabar_hh = 0.5*0.120 // [S/cm^2]
16   // max gNa in soma will be half of axonal value
17 }
18 axon {
19   nseg = 20
20   L = 1000
21   diam = 5
22   insert hh
23 }
24 for i=0,2 dendrite[i] {
25   nseg = 5
26   L = 200
27   diam(0:1) = 10:3 // dendritic diameter tapers along its
28   length
29   insert pas // standard passive current
30   e_pas = -65 // [mv] equil. potential for passive
31   current
32   g_pas = 0.001 // [S/cm2] conductance for passive current
33 }
34
35 # Step 3 - stimulating current.
36 objref stim
37 soma stim = new IClamp(0.5) // put it in middle of soma
38 stim.del = 1 // [ms] delay
39 stim.dur = 0.1 // [ms] duration
40 stim.amp = 60 // [nA] amplitude
41
42 # Step 4 - simulation time course.
43 // set initial conditions
44 dt = 0.05 // [ms] integration time step
45 tstop = 5 // [ms]
46 finitialize(-65) // init membrane potential, state variables,
47   time
48
49 // argument specifies how long to integrate (doesn't reset t
50   to 0)
51 proc integrate() {
52   print t, soma.v(0.5) // show starting time
53   while (t < tstop){
54     fadvance() // advance solution by dt
55     ... // function calls to save or plot results would go
56       here
57     print t, soma.v(0.5) // show present time
58   } // and somatic membrane potential
59 }
```

```
53     ... // statements that change model parameters would go  
54     here  
55 }
```

Listing 2.11 Neuron Code Example.

Critical appraisal: The tool NEURON's strengths clearly lie in the specification of ANN structures and its simulation. The integration with tools such as *NeuroTessMesh* or *NeuroConstruct* is very effective and a prime example of the same. Further, the drag-and-drop-based network construction is intuitive and the dealing with neuronal morphologies is similar to the dealing with process modeling shapes during the construction of process models.

Regrettably, the simulation only considers the processing of electric currents and chemical processes. Any additional context going beyond the injection of currents in neuronal structures is not considered at all. So, modeling language criteria from the process modeling domain, knowledge management, and corresponding simulations cannot be satisfied.

Further, machine learning approaches are not considered at all. Approaches that are not considered are, for example, gradient descent approaches, backpropagation variants, and an efficient data management as required for training and test procedures. So, supervised learning runs and the machine-based adjustment of structures specified cannot be carried out with NEURON.

Although approaches focus on the efficient simulation processing on behalf of parallel computing in NEURON, its processor management is not as effective as *TensorFlow*.

Although the positioning of neuronal structures can consider 3D positions in NEURON, it is limited: first, it is not integrated with its physical surrounding, and second, an augmentation of the real world, for instance, on behalf of AR glasses, is not realized at all.

2.6.2.9 GENESIS

GENESIS was designed as an extensible general simulation system: the **G**eneral **N**eural **S**imulation System. Based on the assumption that the understanding of neuronal networks can be advanced on behalf of the actual anatomy and physiology of the nervous system itself (Bower, 1992; Bower, 2005), GENESIS considers state-of-the-art approaches considering the anatomical and physiological organization of neurons, circuits, and networks (Bower, 1995; Bower and Beeman, 1998) and enables the simulation of corresponding morphologies. Even today, its focus lies on the simulation of many levels of scale, such as subcellular processes, individual

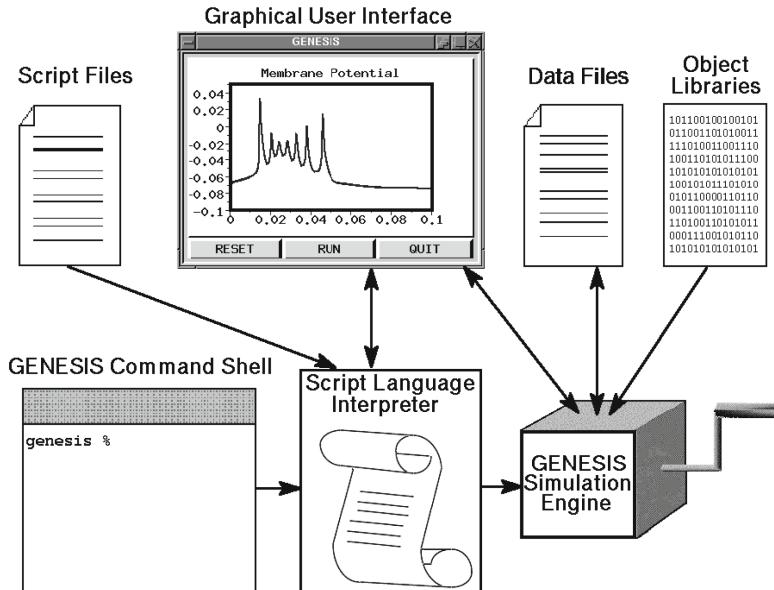


Figure 2.95 Architecture of GENESIS (Beeman, 2005)

neurons, networks of neurons, and neuronal systems (Bower, Cornelis, and Beeman, 2013).

Description: The components of GENESIS are visualized in Fig. 2.95 and explained in the following.

The *simulation engine* of GENESIS can be issued by a command prompt, by the use of simulation scripts, or through its graphical interface allowing users to interactively change simulation parameters in real time. The requirement to be an extensible simulation system is satisfied by the following: first, external *script files* specifying simulations can be considered by the *script language interpreter*; second, external *data files* can be considered by the *simulation engine* so complex neuron models or network connections can be imported as well as exported; and third, *pre-compiled object libraries* can be included by the simulation engine. This enables the following to be specified separately and exchanged easily: individual compartments (spherical and cylindrical); various kinds of channels (synaptically-activated, dendro-dendritic, voltage-activated, and concentration-activated, ligand

gating ionic); and the diffusion of ions within cells (concentration pools, ionic pumps, and buffers) as well as biochemical reactions.

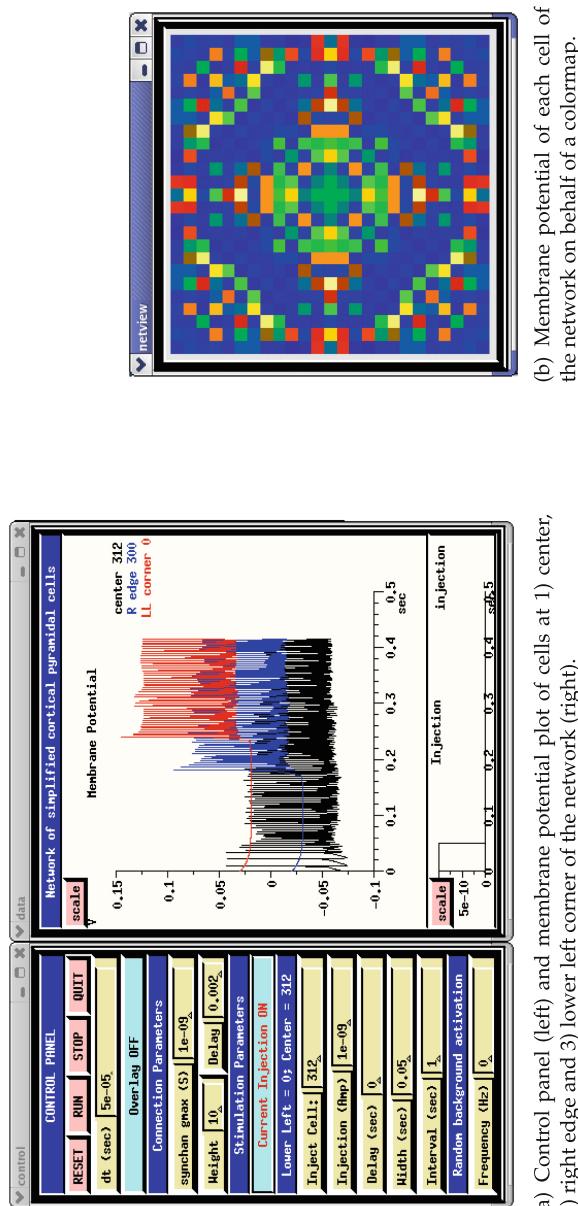
Side facts: Originally, GENESIS was written in C for UNIX platforms and developed by the *Caltech laboratory* (Wilson and Bower, 1989). Since then, it is published under the GNU General Public License and many subsequent releases have progressed the tool's evolution under its community (Genesis-Community, 2018). Now, it also runs under Mac OS and Windows (Bower, Cornelis, and Beeman, 2013) and enables a parallel processing on behalf of a tool variant called PGENESIS (Genesis-Community, 2017). Although currently a third version is under development, the following focuses on the recent stable version (vers. 2.4).

Example: An example can be seen in Listing 2.12, which shows how to create a simulation of a grid of simplified neocortical regular spiking pyramidal cells (according to the GENESIS tutorial of Beeman, 2005). Here, each cell has been coupled with excitatory synaptic connections to its four nearest neighbors.

While only a selection of lines has been visualized in order to give an impression about scripting with GENESIS, within this listing, one can see three sections. First, variables are defined, such as the maximum simulation time and the simulation time step size (lines #4–#5). Further definitions can be included (line #6). Second, functions that have not been prepared in object libraries or script files are defined (line #12). Third, the main simulation is set up. Besides the creation of the simulation clock (line #18) and the creation of views (lines #44–#47), five steps set up the neuronal network. In step 1, network components required have been specified externally. Since they are included (line #21), in step 2, predefined components can be used in lines #24–#26. Step 3 and 4 consider the components in the construction of a network structure, so that step 5 can establish relevant time delays and weights to make them function as network in simulations. Finally, the simulation is started by line #49.

While the code example presented is kept simple because many mechanisms have been included, the specification of individual cells, individual connections, and time-based electric injections and measurements quickly becomes very complex. This is particularly true if the simulation of realistic cells is intended. Graphical interfaces, which were constructed by the code and visualize simulation results, can be found in Fig. 2.96.

While the left interfaces of Fig. 2.96 (a) shows the control panel and allows for the interaction during the simulation, the interfaces on its right visualize the membrane potential of three individual cells in real time. The membrane potential of each cell of the network is visualized in Fig. 2.96 (b) on behalf of a colormap.



- (a) Control panel (left) and membrane potential plot of cells at 1) center, 2) right edge and 3) lower left corner of the network (right).
- (b) Membrane potential of each cell of the network on behalf of a colormap.

Figure 2.96 Genesis Tutorial Example (Beeman, 2005)

```
1 // =====
2 // Variable definitions
3 // =====
4 float tmax = 0.5      // simulation time
5 float dt = 50e-6      // simulation time step
6 include synapseinfo.g
7 ...
8
9 // =====
10 // Function definitions
11 // =====
12 ...
13
14 // =====
15 // Main simulation section
16 // =====
17 # set the simulation clock
18 setclock 0 {dt}
19
20 # Step 1: Create any prototype channels, compartments, etc.
21     that will be used to build the cells.
21 include protodefs.g // This includes library of cell
22     components
23
24 # Step 2: Create a prototype cell, coupled to an excitatory
25     synchan and a spikegen.
26 readcell {cellfile} /library/cell
27 setfield /library/cell/{synpath} gmax {gmax}
28 setfield /library/cell/soma/spike thresh 0 abs_refract 0.004
29     output_amp 1
30
31 # Step 3 - make a 2D array of cells with copies of /library/
32     cell
33 createmap /library/cell /network {NX} {NY} -delta {SEP_X} {
34         SEP_Y}
35
36 # Step 4: Now connect them up with planarconnect.
37 planarconnect /network/cell[]/soma/spike /network/cell[]/{
38     synpath} \
39     -relative \
40     -sourcemask box -1 -1 1 1 \
41     -destmask ellipse 0 0 {SEP_X*1.2} {SEP_Y*1.2} \
42     -desthole box {-SEP_X*0.5} {-SEP_Y*0.5} {SEP_X*0.5} {SEP_Y
43         *0.5} \
44     -probability 1.1
45
46 # Step 5: Use the planardelay and planarweight commands to
47     provide appropriate axonal delays and synaptic weights to
48     the connections.
49 planardelay /network/cell[]/soma/spike -fixed {prop_delay}
50 planarweight /network/cell[]/soma/spike -fixed {syn_weight}
51
52 # Create simulation views
53 include graphics // include the graphics functions
54 make_control // make the control panel
55 make_Vmgraph // make the graph to display soma Vm
56 make_netview // make the newview
57 ...
58 reset
```

Listing 2.12 Genesis Code Example.

Critical appraisal: The tool GENESIS clearly has its strength in the very powerful construction of neuronal morphologies, particularly in the use of ion specific channels and their use in simulations. Many wizards, such as *Neurokit* and *Kinetikit* enable the construction, running, and visualization of single cell models and biochemical reactions with little or no programming effort. The interactive modification of simulation parameters and the mouse-based injection electrode positioning into a graphical representation of a cell is a prime example of the above and greatly supports the usability as simulation and modeling tool. The integration with tools such as *NeuroTessMesh* or *neuroConstruct* is very effective and integrates 3D based neuronal models. So, common simulation tool aspects can be satisfied with a focus on ANN. Regrettably, an integration with modeling elements coming from the process domain is not considered and an AR integration is missing as well.

Although GENESIS considers individual neuronal structures, the corresponding modeling shapes are not provided. Further, the dealing with various abstraction levels is difficult since a common, hierarchical view is missing. Since a perspective-oriented possibility to neglect irrelevant information by views is not available at all, graphical representations and control mechanisms quickly become overwhelming.

Although GENESIS allows an automated parameter search, which considers the five approaches called brute-force parameter search, conjugate-gradient descent parameter search, genetic algorithm parameter search, simulated annealing parameter search, and stochastic parameter search (Bower and Beeman, 2003), the parameter modification is not as flexible and powerful as programming libraries providing common machine learning approaches. So, criteria focusing on data management, training, and test procedures and approaches required by machine learning approaches cannot be satisfied.

2.6.2.10 Critical Appraisal

The critical appraisal of ANN and DL tools is based on the following: first, approaches of a tool evaluation and selection are appraised; then, ANN and DL tools are appraised; and finally, an ANN tool-specific facet is identified.

Tool Criteria: Most approaches to evaluate ANN tools are based on a set of criteria. Since those were defined in order to identify suitable ANN tools with regard to a certain evaluation purpose, which was not the construction or realization of a CoNM, an appropriate CoNM selection system, per se, is still missing. Hence, till date, ANN tools have neither been evaluated to be the foundation for the CoNM nor have they been evaluated to be a realization of the CoNM.

In order to decide whether current evaluation frameworks can at least partly identify a CoNM foundation or which criteria is suited to be part of a CoNM criteria catalog, the following appraises criteria category-wise:

1. Since the CoNM is intended to be carried out by commonly known tools, tool usage criteria focus on aspects which are accepted in the deep learning community. Criteria trying to apply accepted concepts from the ANN domain in the ANN process domain are highly relevant.
2. Since the CoNM is intended to be carried out by commonly known tools, technical concept criteria of ANN and DL tools focus on technical aspects which are accepted in the deep learning community, simplify the implementation, and ensure the connection to practices and standards. These are, therefore, highly relevant to be part of the CoNM criteria catalog as well.
3. Although the selection of tools obviously must consider the individual situation of the tool user (similar to process modeling tools), user-specific criteria are not yet considered in the deep learning community.
4. From the perspective of the tool vendors, a CoNM is to be integrated in the company's business model. Since those are modifications of the general concepts presented here, they are not required for the CoNM criteria catalog designed herein.

Since no single approach provided criteria regarding the use of techniques and concepts from the process domain, which is an essential part of the ANN process domain, none of provided evaluation frameworks is able to at least partly identify a CoNM foundation from the perspective of the ANN and DL domains. Further, this means that the deep learning domain has not yet been brought together with process modeling techniques. A research gap becomes visible here, which asks for the following: first, the construction of a CoNM because anything similar is missing so far; second, a CoNM tool selection system that extends common criteria and is able to evaluate current tools; and Third, an analysis of currently available deep learning tools for the use of CoNM relevant aspects such as neuronal techniques, specific modeling objects, and common ANN tool criteria.

ANN Tools: Since ANN tools considered do not integrate with the process modeling and simulation domain, corresponding modeling language criteria cannot be satisfied at all. ANN tools only consider modeling objects inspired by traditional machine learning and neuronal network research. While each realizes a highly specific purpose, only some provide a graphical visualization and allow a modeling on behalf of modeling items.

First, there exist tools particularly supporting machine learning on the basis of artificial neuronal networks. Here, one can find *PyBrain*, *Theano*, *Caffe(2)*, *TensorFlow* incl. *TensorBoard* and *LUCID*. While all of the tools realize an ANN modeling on the basis of more or less laborious programming code, only the latter five provide graphical visualizations of the network structure and functioning. In any case, these only focus on operations so that an efficient computation on different computing units can be realized. The network structure, its processual tasks and input and output of a neuron-specific level are not considered by any programming tool. Since visualizations are derived from the codebase, an interplay in the sense of graphically modeling network structures is not possible. Further, these tools neither consider the plausible physical positioning of neurons within a network structure nor the physical constitution of its compartments.

Second, there exist tools particularly focusing on the 3D modeling of neurons. Here, the plausible physical positioning of neurons within a network structure and the physical constitution of its compartments is considered. Examples include the tools *neuroConstruct* and *NeuroLOTs*, with *NeuroTessMesh*. Here, a learning in the sense of machine learning, which can be seen as a kind of approximation of biological models, is not realized. The physical modeling for a realistic distribution of currents is the central element. Regrettably, neither the physical positioning within the real world nor the augmentation of the real world is realized. Thus, interpretable results with regard to typical physical meanings coming from the process domain (neurons representing obstacles, bordering spatial processes, or functioning as local machines) cannot be derived. The constitution focuses only on the use of biological and neuromedical results and the process domain is not considered at all.

Third, there exist tools particularly focusing on the simulation of neuronal structures. Here, one can find *GENESIS* and *NEURON*. Since these focus on the biologically-plausible and neuromedically-correct simulation, neither a learning in the sense of machine learning nor a modeling in the sense of process modeling is considered at all. Although simulation tools can built on realistic 3D models of neurons and can import files from *neuroConstruct* and *NeuroTessMesh*, the aforementioned limitation with regard to interpretable results regarding typical physical meanings coming from the process domain are not overcome by this kind of tools. Hence, simulations coming from the process domain and particularly business process relevant simulations cannot be carried out with this kind of tool.

Considering the three kinds of tools identified, no approach is able to at least partly integrate with the process domain. Further, modeling tool criteria are not satisfied at all. Simulations focus on the realization of biologically-plausible and neuromedically-correct model use. Modeling objects are rather limited and their use in the sense of an interactive modeling is not realized by any approach. A research

gap becomes visible here, which asks for the following: first, the construction of a CoNM because anything similar is missing so far; second, a CoNM tool that satisfies common ANN and DL tool criteria and is able to consider the process domain and modeling and simulation tool criteria; and third, an analysis of currently available deep learning tools for the use of CoNM relevant aspects such as neuronal techniques, specific modeling objects, and common ANN tool criteria.

Facit: Neither ANN tool evaluation frameworks nor ANN and DL tools consider the integration of the process domain and the ANN domain. Criteria collections do not provide process modeling and simulation criteria and tools do not realize the aforementioned criteria. Here, a research gap within the ANN domain becomes visible.

2.6.3 Conclusion About Tools and Libraries

The conclusion about tools and libraries considered is based on the critical appraisals of process modeling and simulation tools (section 2.6.1.9) as well as the critical appraisal of ANN and Deep Learning tools (section 2.6.2.10). First, approaches of a tool evaluation and selection are concluded. Then, process modeling and simulation tools are concluded. Finally, a tool-specific conclusion is identified.

Tool Criteria: Neither process modeling and simulation tool evaluation frameworks nor ANN and DL tool evaluation frameworks consider the integration of the process domain and the ANN domain. Criteria collections from the process domain do not provide ANN and Deep Learning criteria and criteria collections from the ANN domain do not provide process modeling and simulation criteria.

Since no single approach provided criteria regarding the use of techniques and concepts from both domains, which is an essential part of the ANN process domain, none of provided evaluation frameworks is able to identify a CoNM foundation at all. Further, this means that the deep learning domain has not yet been brought together with process modeling techniques, and vice versa. A research gap within the intersection of the process domain and ANN domain becomes visible here, which asks for the following: first, the construction of a CoNM because anything similar is missing so far; second, a CoNM tool selection system that extends common criteria collections and is able to evaluate current tools; and third, an analysis of currently available deep learning tools as well as process and simulation tools for the use of CoNM relevant aspects.

Tools: Neither process modeling and simulation tools nor ANN and DL tools consider the integration of the process domain and the ANN domain. Tools from the process domain do not realize ANN and Deep Learning criteria and tools from the ANN domain do not realize process modeling and simulation criteria.

Since no single approach is able to integrate with both domains, which is an essential part of the ANN process domain, none of provided tools is able to stand as a CoNM foundation by themselves. A research gap within the intersection of the process domain and ANN domain becomes visible here, which asks for the following: first, the construction of a CoNM because anything similar is missing so far; second, a CoNM tool that satisfies common ANN and DL tool criteria from the ANN domain as well as further considers modeling and simulation tool criteria from the process domain; and third, an analysis of currently available modeling and simulation tools as well as ANN and deep learning tools for the use of CoNM relevant aspects.

Conclusion: Since neither tool evaluation frameworks have considered criteria from the ANN domain and the process domain nor have tools realized the aforementioned criteria, a research gap becomes visible in the ANN process domain. This asks for the following three: first, the construction of a CoNM; second, the design and demonstration of a CoNM selection and evaluation framework; and third, the implementation and demonstration of a CoNM tool.

2.7 Problem Analysis and Research Gap

Following the methodology selected in section 3.2.1, the problem analysis can be realized by an SLR as it identifies concepts out of the literature and considers them with regard to the problem statement defined. The use of concrete analysis aspects supports the identification of missing concepts and, therefore, operationalizes the research gap.

SLR method: The research gap was confirmed by an SLR in accordance with the PRISMA statement (Liberati et al., 2009; Moher et al., 2009). Although it originally comes from the medical domain, it can be applied in software engineering context (Budgen and Brereton, 2006; Kitchenham et al., 2009) focused upon here. It was carried out as described in section 3.3. Hence, the following presents results being structured by the SLR components.

Objective specification: The SLR was carried out in order to create an overview of relevant domains. Further, the objective of the SLR was the identification of relevant concepts such that unnecessary work can be avoided and a conceptual base for the creation of the CoNM can be established. Finally, it intended to confirm the research gap.

Literature research questions: Questions addressed by the study are:

1. **LRQ:** How much activity has there been regarding process optimizations by artificial neuronal networks?
2. **LRQ:** What research topics are being addressed with regard to the intersection of ANN (including deep learning) and process modeling (including simulation and optimization)?
3. **LRQ:** Who is leading neuronal process optimization research?
4. **LRQ:** What are the limitations of current research?

Limitations of LRQ 4 have been addressed by the following issues:

- 4.1. **LRQ:** Were the research topics limited?
- 4.2. **LRQ:** Is there evidence that the use of ANN for process optimization is limited due to a lack of functioning?
- 4.3. **LRQ:** Is the quality of contributions appropriate following a clear mathematical formalization?
- 4.4. **LRQ:** Is literature contributing to practice by defining practice guidelines?

Search process: The search process was a manual review of conference proceedings and journal papers, executed by the author using ISWorld's top 50 MIS journals and IS conferences (Levy and Ellis, 2006) via the following databases: Elsevier (ScienceDirect), IEEE (Comp Soc & Xplore), ACM (Digital Lib), Google Scholar, and Full Text Web Access (Fee and Free). Table 2.2 shows the 22 journals and 12 conferences being considered.

Further, peer-reviewed contributions were considered that follow the department-specific philosophy on processes and knowledge management of the “Department of Business Informatics, esp. Processes and Systems” of the University of Potsdam.

Table 2.2 SLR Journal and Conference Selection

			Literature Vendor (Database)				
Type	No.	Name	Elsevier (ScienceDirect)	IEEE (Comp Soc & Xplore)	ACM (Digital Lib)	Full Text Web Access (Fee)	Full Text Web Access (Free)
Ranked MIS Journals (Based on ISWorld)	1	Communications of the ACM				✓	
	2	Artificial Intelligence	✓		✓		
	3	IEEE Transactions (various)		✓	✓		
	4	Decision Support Systems	✓				
	5	IEEE Software		✓			✓
	6	Information & Management	✓	✓			
	7	ACM Transactions on DB Sys			✓		
	8	IEEE Transactions on Software Eng		✓			
	9	ACM Transactions (various)			✓		
	10	J of Computer and System Sci	✓				
	11	Communications of the AIS			✓		
	12	IEEE Trans on Sys, Man, & Cyb		✓			
	13	ACM Computing Surveys			✓		
	14	Journal of the AIS			✓		
	15	IEEE Transactions on Computers		✓			
	16	IEEE Computer		✓			
	17	Informations Systems	✓				
	18	Journal of the ACM			✓		
	19	Computers & Operations Research	✓				
	20	Journal of Strategic IS	✓				
	21	ACM Transactions on IS			✓		
	22	Informing Science					✓
Ranked and Non-Ranked IS Conferences (Based on Hardgrave and Walstrom, 1997)	1	International Conference on Information Systems (ICIS)		✓			
	2	Hawaii International Conference on System Sciences (HICSS)		✓		✓	
	3	International Federation for Information Processing (IFIP)		✓	✓		
	4	International Conference on Decision Support Systems (DSS)		✓			
	5	Decision Science Institute (DSI) - National Conference				✓	
	6	Society of Information Management (SIM) Conference	✓				
	7	International Association for Computer Information Systems (IACIS) Conference (Proceedings published in Issues in Informations Systems)				✓	
	8	Academy of Management (AOM) Conference			✓		
	9	Decision Science Institute (DSI) - Regional Conferences			✓		
	10	International Academy of Information Management (IAIM) Conference			✓		
	11	American Conference on Information Systems (AMCIS)			✓		
	12	Information Systems Education Conference (ISECON)				✓	
	13	Institute of Electrical and Electronics Engineers (IEEE) National Conferences	✓				
	14	Informing Science + Information Technology Education (InSITE) Conference				✓	

Search terms: The SLR considered combinations of the following search words:

- w – Process Domain:
“Process”, “Business Process”, “Knowledge”

- x – ANN Domain:
“Deep Learning”, “Artificial Neural Networks”
- y – Objective:
“Modeling”, “Simulation”, “Optimization”, “Prediction”
- z – Purpose:
“Interpretation”, “Understanding”, “Comprehension”, “Controlling”, “Monitoring”, “Extraction”, as well as the tooling synonyms “Tools”, “Application”, “Framework”, “Compiler”, “Library”, “Environment”, “Tool Selection Framework”.

Considering alternative grammar specifics, the following search words were further included:

“Process Optimisation” and “Process Optimization”,

“Artificial Neuronal Networks” and “Artificial Neural Networks”,

“Modelling” and “Modeling”.

The identification of terms was part of the SLR realization and its repetitive realization was in accordance with section 3.3. So, search terms were combined by Boolean operators as follows:

$$w \text{ AND } x \text{ AND } (y \text{ OR } z)$$

Keyword searching: The keyword searching was realized by *backward searching* and *forward searching* (Levy and Ellis, 2006): the first included a *backward reference search*, *backward author search*, and *previously used keyword search*, while the latter included a *forward reference search* and a *forward keyword search*. This enriched both the identification of SLR search words and the identification of relevant literature.

Inclusion criteria: The SLR only considered peer-reviewed articles as the body of knowledge where a combination of at least two terms were an element of the article (*two search word combinations*). These focused on the title, keywords presented, abstract and the full text. Three-word combinations have been considered as objects of investigations (OoI) having been described in detail according to Bloom et al. (1956). These also focused on the title, keywords presented, abstract, and the full text.

Exclusion criteria: Articles which were not peer-reviewed, and informal literature reviews (no defined research question, no underlying methodology, or no demonstration and evaluation) were excluded from the SLR in the first step. Further, articles focusing on the efficient task distribution across CPU and GPU have been excluded as these do not support the CoNM focus. Exceptions were made only in regard to company-driven market research studies of process modeling tools and highly innovative ANN interpretation approaches and duplicate reports of the same study were dismissed beforehand. Further, articles that focused on wrong contexts were removed, an example of which can be found in the topic of “email security level classification”.

Quality assessment: Considering the quality criteria of the York University, Center for Reviews and Dissemination (CRD) Database of Abstracts of Reviews of Effects (DARE), the quality was assessed on behalf of the following questions:

1. **QA:** Were inclusion/exclusion criteria reported?
2. **QA:** Was the search adequate?
3. **QA:** Were the included studies synthesized?
4. **QA:** Was the quality of the included studies assessed?
5. **QA:** Are sufficient details about the individual included studies presented?

In addition to the previous exclusion criteria, entries have been excluded as and when these criteria have not been satisfied. According to Liberati et al. (2009), Fig. 2.97 presents the final flow diagram of the SLR conducted.

Here, one can see the iterative literature selection. Finally, 1,119 records have been analyzed in detail, while 38 texts have been analyzed as individual OoI. These represent various approaches issuing the same ideas and modifications. Since approaches mainly deal with ideas conceptually, an empirical examination by remarkable high test persons has not been realized by any contribution. Some provide a workshop-based validation so that generally, the technical feasibility of an idea has been focused. For relevant entries, data has been extracted as follows.

Data extraction: Data extracted from each study are the following:

- The source (journal, conference, practitioner, department) and full reference.
- Classification of the article type (concept, market overview, SLR, literature overview, meta-analysis, study); Scope (process domain, ANN domain or ANN process domain).

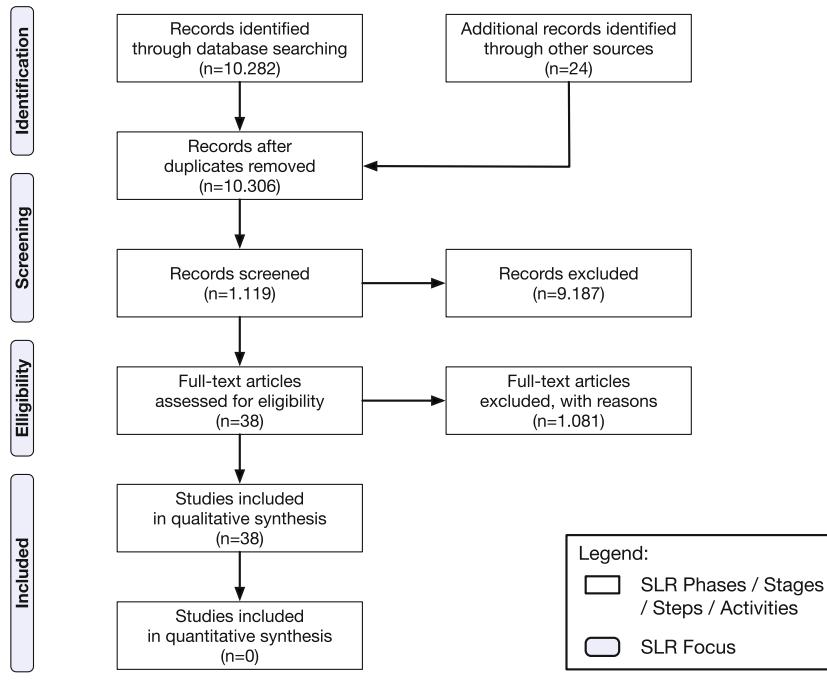


Figure 2.97 SLR Flow Diagram

- Topic and main topic area (modeling language, knowledge representation, simulation, optimization, learning, tooling).
- The author(s); their institution, the country where it is situated.
- Summary of the article including the main research questions and the answers.
- Research issues in regard to limitations.
- Quality evaluation.

The author extracted the data and BPM students of the University of Potsdam confirmed the extraction. The resulting *two-term contributions* are presented and critically appraised in sub-sections of section 2. Here, how the contribution exactly can be placed in the context of existing work, the so-called *body of knowledge*, becomes clear.

Searching stop: The search process was finalized when it was discovered that new contributions only introduce familiar concepts (Leedy and Ormrod, 2005).

Body of knowledge: The SLR can be placed in the context of the following works. An important area of *knowledge management* research consists of the identification of tacit knowledge and looks for ways that support the ability to turn knowledge into effective action. The corresponding research questions make the identification of organizational practices and systems to transfer knowledge among organizational units subject to discussion. The CoNM can be placed in this context since it aims to deal with tacit knowledge and tries to turn it into effective action.

A great focus of the *deep learning* and *artificial neuronal network* research lies, first, on the approximation of a system behavior and second, on the increase of the interpretability of artificial neuronal networks, which is supported by feature visualization. Here, research questions make the purpose-related model creation in specific contexts with huge, time-dependent networks and the algorithmic processing subject to discussion. The CoNM can be placed in this context because of the following two. First, it aims to approximate the behavior of an organization, including its entities, and second, it aims to support the interpretability of the resulting ANN.

The area of *business process management* draws attention to the creation of process models, their use in simulations (simulation models), and their management, which includes process optimizations. The corresponding research questions deal with the design of modeling languages, their tool-based use, and potentials carried out. The CoNM can be placed in this context since it aims to apply biological models to the creation and optimization of process models. Further, it intends to apply them in process simulations.

The area of *architecture management* draws attention to the design and conceptualization of information systems, which includes their support for business contexts. Research questions of this domain focus on the identification of required components, their company-wide interplay, and best manifestation in economic environments. The CoNM can be placed in this context since it aims to design a framework for the modeling on behalf of IS dealing with ANN.

The research gap addressed here focuses on the intersection of research areas presented and reflects model creation by neuronal techniques.

Results: Contributions resulted by *three search word combinations* are visualized in Table 2.3, which forms the concept matrix augmented with dimensions of the *research scope* as units of analysis (Webster and Watson, 2002).

Table 2.3 Concept Matrix

Articles	Concepts											
	Modeling Language		Knowledge Representation		Simulation		Optimization		Learning		Tooling	
Unit of Analysis	P	ANN	P	ANN	P	ANN	P	ANN	P	ANN	P	ANN
Petri, 1962	✓				~							
Scheer, 2002	✓											✓
Booch, Rumbaugh and Jacobson, 2005	✓											
White, 2004	✓											
Karagiannis and Telesko, 2000	✓	✓										
Heisig, 2002, 2006	✓	✓										
Gronau et al., 2003	~	✓										✓
Gronau, Pogorzelska, 2012	✓	✓			~							✓
Borshev et al., 1997					✓					~	✓	~
Gilbret and Gilbret, 1921					~							
Forrester, 1961					✓							
Fayol, 1900 (Financial Times, 1994)							✓			~		
Masaaki, 1986							✓			✓		
Scheer, 1997							✓			~		
Argyris and Schön, 1978							✓			✓		
Bateson, 1972							✓			✓		
Keller, Nüttgens and Scheer, 1992	✓	✓			~						✓	
Drawehn et al., 2014	✓	✓			~						✓	
Gronau, 2016	✓	✓									✓	
Decker, Overdick and Weske, 2008	✓				~						✓	
Müller and Vignaux, 2002					✓						✓	
Molder, 1974	~				✓	✓	~			✓		~
Schaul et al., 2010							✓			✓		✓
Bergstra et al., 2010							✓			✓		✓
Jia et al., 2014	~						✓			✓		✓
Abadi et al., 2015	~						✓			✓		✓
Gleeson et al., 2007	~						~			~		✓
Garcia-Cantero et al., 2017	~						~					
Hines and Carnevale, 1997							✓					
Bower and Beeman, 1998							✓					
Wongsuphasawat et al., 2018	✓						~			~		✓
Gleeson et al., 2010	✓						~					
Cannon et al., 2014	✓						✓			~		~
Kruse et al., 2015					~							
Erhan et al., 2009					~							
Zeiler and Fergus, 2013					~							
Nauk, Klawonn and Kruse, 1997					~							
Grum and Gronau, 2018	✓	✓	✓	✓	~	~	~	~		~		

Here, one can see articles and their relation to relevant concept areas indicated by check marks. The tilde symbol indicates whether articles at least partly satisfy criteria or consider them in prospect. Since options of the *research scope* have been considered as unit of analysis, it becomes clear which concept area an article can be related to. Further, it becomes clear which domain an article can be mapped to. Hence, it is easier to detect when an article strays outside the scope of its domain.

Considering all articles, it can be seen that no all-encompassing article can be related to all relevant concepts. Further, it can be seen that articles are either considered within the process domain or the ANN domain. No article strays outside the

scope of this domain and can be considered in both domains. Thus, it becomes clear that each domain (process and ANN) has its own meaning for knowledge, modeling, simulation, optimization, learning, and tooling. To the best of our knowledge, only Grum and Gronau (2018a) issue a first junction of the same. Therefore, the definition of an *ANN process domain* as the intersection of the *process domain* and *ANN domain* is ideal.

LRQ outcome: Regarding LRQ 1 (“How much activity has there been regarding process optimizations by artificial neuronal networks?”), one can find a great amount of activity in pioneering works about ANN techniques from the 1950s onwards (McCulloch and Pitts, 1943; Hebb, 1949; Rosenblatt, 1958; Zadeh, 1965; McCulloch and Pitts, 1988; Bishop, 1995). Although an AI winter can be found from 1969 to 1986, a trend of activities can be identified because of, first, a performance boost caused by new techniques such as Convolutional neuronal networks (LeCun et al., 1998) and Long Short-Term Memory networks (Hochreiter and Schmidhuber, 1997a), as well as, second, a computing power increase (Schwaiger and Steinwendner, 2019, p. 255, 274). Although, with these greater possibilities, ANNs can become much deeper, techniques primary have been applied for simple prediction problems focusing on only one process step. Examples include process parameter modifications, such as molding processes (Shen, Wang, and Li, 2007) or chemical reactions (James, Lam, and Li, 2011), so that process conditions can be adjusted (Shabani and Mazahery, 2012). Also, modifications on behalf of the identification of relationships (Teran and Grotewold, 1996), the determination of an optimized input to produce a target output (Davis and Gasperi, 1997), modifications because of predictions (Chen, Ramaswamy, and Alli, 2001), and meta-modeling of interrelated components in simulations (Chambers and Mount-Campbell, 2002b) are just some of the representatives and activities around process optimization by ANN. Till date, certain predictions or ANN outputs are generated by ANN techniques, following which, corresponding measures are derived and applied, for instance, by human process participants. As the use of the ANN results goes beyond the neuronal context, its consideration is limited to the ANN prediction task.

Regarding LRQ 2 (“What research topics are being addressed with regard to the intersection of ANN (including deep learning) and process modeling (including simulation and optimization)?”), one can identify isolated attempts that operate in their own domain rather than joint concepts. First associations are shown in the following:

First, focusing on modeling, both ANN models as well as process models are visualized by graphical modeling languages. While graphical ANN models are visualized by ANN modeling languages, these build on a code visualization. An inter-

active and cooperative modeling, as it is known from the process modeling domain, has not yet been realized. Process modeling, on the other hand, does not include technical mechanisms and these operate on a graphical base only. Further, neither an ANN modeling on the base of process modeling techniques nor a process modeling on the base of ANN techniques have been realized.

Second, focusing on simulations, process simulations are mainly limited to techniques from the process domain. Intersecting research topics realize the integration of isolated ANN systems and simple processes for simulation purposes. On the other hand, further extension, namely the integration of complex ANN process participants, that interacts within the simulation of complex processes is still missing. Further, ANN simulations majorly focus on ANN structures. Integrating various instances of ANN systems in one joint simulation system, or the consideration of ANN to be a simulation system, and representing joint economic contexts have not been addressed before.

Third, focusing on optimization, ANN techniques are considered as approximation tools for simple and individual process steps. They have not been considered as optimization tools for process optimizations on complex ANN model-bases. A deep process model optimization has not been realized either. On the other hand, process optimization techniques only focus on the process context and have not been used for ANN training purposes. Even though working tools are available, a joint process understanding is missing so far. Also, a bidirectional concept exchange among disciplines has not yet been realized.

Regarding LRQ 3 (“Who is leading neuronal process optimization research?”), one must identify that a neuronal process modeling, simulation, and optimization, as it is intended by the CoNM, has not been realized before. Hence, a single party that is leading this research field cannot be identified. As one focuses on the leading instances of its intersecting topics, one can find five expert clusters: first, capable ANN techniques as well as modern interpretation approaches at the University of Linz with Prof. Hochreiter, the Swiss research institute for AI called IDSIA, and the Google Brain Team; second, powerful process modeling tools at the Scheer Group with Prof. Scheerer; third, capable knowledge modeling approaches and tools at the University of Potsdam with Prof. Gronau; fourth, a varied range of process optimization clusters following individual philosophies; and fifth, a simulation competence cluster at the Saint Petersburg Polytechnic University around Borshchev, Karpov, and Roudakov.

The LRQ 4 (“What are the limitations of current research?”) has been detailed by four questions. Hence, the following presents an answer as per each sub-question. Regarding LRQ 4.1. (“Were the research topics limited?”), one can find various

limitations regarding the research topics of modeling, simulation, and optimization in working with ANNs:

First, the possibility to interpret ANN structures limits ANN technique applications in a broad context. So far, an automated and plausible identification of a neuron's meaning within the networking structure is not possible at all. With regard to the research topics, this means that ANN systems only can be considered as black boxes.

Second, a guarantee for a successful training with ANN cannot be given. Here, the dealing still is considered as "black magic" and a debugging of failed training attempts is mainly based on experience and interpretation.

Third, working with models of biologically-plausible mechanisms is limited since ANN structures simplify mechanisms. As they do not reflect all biological insights available (cf. section 2.5.1), a transfer to further contexts is limited to the abstractions implemented. Examples for omitted mechanisms (cf. Fig. 2.1) can be found in the neuron's compartments, such as the *mitochondrion*, *endoplasmic reticulum*, and *golgi apparatus*, which are not considered by the programming libraries or tools analyzed. Further, the spatial position of biological compartments is not considered in programming libraries either. Since the spatial positioning is essential for organizational entities in business processes, a bidirectional transfer among domains is limited. This includes the use of AR techniques for the spatial interpretation of neuronal processes in business contexts.

Regarding LRQ 4.2. ("Is there evidence that the use of ANN for process optimization is limited due to a lack of functioning?"), besides a lack of computing power or infrastructural issues, it can be declared that mainly because of a lack of interpretability of ANN structures, the use of DL structures is limited to use within economic contexts (i.a. Kruse et al., 2015, cf. section 2.5.7). Moreover, if essential decisions are intended to be supported or even carried out by the CoNM, the trustworthiness in algorithmic approaches is as a hygiene factor essential for its acceptance. As a clear interpretation of ANN structures and its dynamic effects is a requirement for unfolding the full potential of the CoNM, the union of ANN techniques and business process interpretations is challenged, especially as an adequate interpretation approach is missing so far.

Regarding LRQ 4.3. ("Is the quality of contributions appropriate following a clear mathematical formalization?"), one must notice that contributions provide formalizations at different levels.

The first category refers to modeling languages that are able to provide a mathematical formalization by means of visualizations. Here, only some contributions of modeling language-based approaches satisfy a strictly formalized mathematical definition (Gleeson et al., 2010; Cannon et al., 2014; Wongsuphasawat et al.,

2018). Most representatives must be categorized as semi-formal modeling language approaches (Gilbreth and Gilbreth, 1921; Forrester, 1961; Borshchev, Karpov, and Roudakov, 1997; Pogorzelska, 2012). As the CoNM intends to unit the ANN domain and process domain, this challenges its construction. Nonetheless, the strictly formalized modeling languages are adequate to work with and semi-formal modeling languages can be adequate, if a formalization is complemented beforehand.

The second category refers to programming libraries, that provide implementations of mathematical formalizations. Although the mathematical formalizations are not explicitly and transparently visualized, the formalizations can be examined on a codebase. Here, various programming libraries can stand, at the very least, as a common technical base (Bergstra et al., 2010; Schaul et al., 2010; Jia et al., 2014; Abadi et al., 2015). If the programming library enables the code modification and extension, the quality is adequate for the CoNM construction.

The third category refers to tools providing a graphical user interface and having mathematical formalizations implemented. Here, mathematical formalizations are neither explicitly visualized, nor is their codebase always available. They provide, at the very least, a common technical base (Hines and Carnevale, 1997; Bower and Beeman, 1998; Gleeson, Steuber, and Silver, 2007; Garcia-Cantero et al., 2017). As the fewest tools enable the modification and extension of underlying mechanisms, the quality is adequate to work with.

Regarding LRQ 4.4. (“Is literature contributing to practice by defining practice guidelines?”), one can recognize that the process domain particularly provides numerous methods, proceedings, and guidelines to realize process and knowledge management (Allweyer and Scheer, 1995; Gronau, 2009; Grum and Gronau, 2017). Further, these are accompanied with numerous practical case studies. On the other hand, ANN techniques do not provide sophisticated practice guidelines. Although generic approaches are available, such as CRISP (Schwaiger and Steinwendner, 2019, p.275) or the method described by Lämmel and Cleve (2012, p.199), ANN techniques often demand individual proceedings. Here, proceedings only show the functioning of an idea by case studies rather than stating a wide collection of examples. Further, only some provide a corresponding database (Szigeti et al., 2014; Sarma et al., 2018; Bowen et al., 2019) or repository (BAIR, 2018a; BAIR, 2018c; Cannon et al., 2018; Google-Brain-Team and Community, 2018b; Genesis-Community, 2018; GMRV and Community, 2018b; Google-Brain-Team and Community, 2018a; IDSIA, 2018; LISA lab and Team, 2018a). In summary, literature mainly contributes towards practice with guidelines in the process domain. In contrast, ANN domain contributions are lacking in these guidelines.

Problem Analysis: The following summarizes and synthesizes insights which already have been presented in individual critical appraisals of the previous subsections and the concept matrix shown before. Since those refer to limitations and issues, the following points characterize the research gap and serve as a problem analysis.

- 1) Concept-centric objects of research clearly focus on isolated domains which are the modeling as well as the simulation of processes, the codification as well as the organization of knowledge in knowledge management systems, and the optimization of business processes with non-ANN techniques as well as the modeling and use of ANN in ANN-specific learning problems.
 - ⇒ Faced with highly-specialized, domain-isolated concepts, no object of research addresses the process optimization by ANN.
- 2) The process domain provides numerous definitions from various domains (philosophy, computer science, economy, business informatics, etc.). A cascade of definitions ranges from generic to concrete, from philosophical to economical, and from ancient to modern.
 - ⇒ Faced with a great heterogeneity of process definitions, no definition considers neuronal process definitions.
- 3) Numerous meta-models for model notations and various modeling languages provide a great number of modeling perspectives. Many of them consider concrete contextual relations, such as business, biology, and chemistry, among others. Further, they differ in the number of abstraction layers. Some modeling languages are not even connected to meta-models.
 - ⇒ Faced with a great heterogeneity of modeling perspectives, no meta-model or modeling language considers neuronal process perspectives.
 - ⇒ Faced with a great heterogeneity of modeling abstraction layers, modeling languages are difficult to compare. Further, it is complicated to integrate modeling languages having different abstraction layers.
 - ⇒ Faced with modeling languages not connected to any meta-model, representatives are restricted to the provided perspectives. A generalization cannot be realized here at all. A comparison with similar modeling languages is also

very difficult here. Further, an integration with other modeling languages is very complicated.

⇒ Neuronal modeling languages providing neuronal perspectives such as *NeuroML* focus only on the structure and dynamics of neuronal networks (Cannon et al., 2014; Vella et al., 2014). Since a contextual relation and a relation to a common meta-model are missing, modeling language comparisons or integrations are increasingly difficult.

- 4) Enterprise architecture models are used for the description and integration of process models and information systems. This demands for the consideration of different technical components. Therefore, the literature provides numerous enterprise architecture management (EAM) approaches. Each draws attention to individual aspects, demands different perspectives, and intends to be sufficient for the whole enterprise.

⇒ Faced with a great heterogeneity of architecture models, no model describes and integrates process models and information systems of neuronal techniques or provides ANN-relevant architecture components.
- 5) A great base of modeling languages and taxonomies can stand as a foundation of new modeling notations. Specific modeling notations can be found for process modeling, business process modeling, knowledge modeling, and first drafts of ANN modeling.

⇒ Faced with a great heterogeneity of modeling objects, no language considers both process or business process elements and neuronal elements.
- 6) Knowledge management concepts deal with the identification and management of knowledge. Further knowledge codification approaches can be found in information system research.

⇒ Faced with a great heterogeneity of knowledge representation approaches, no concept is able to explain neuronal representations.

⇒ Knowledge management concepts explicitly focus on knowledge represented. Tacit knowledge is only considered on a qualitative basis.
- 7) Knowledge management concepts are motivated from various perspectives and aim to efficiently manage knowledge.

- ⇒ Faced with a great heterogeneity of knowledge management concepts, no concept follows biological models and considers neuronal mechanisms.
 - ⇒ Further, knowledge management concepts are used without the use of AI.
- 8) Computer simulations are applied to various domains. First attempts that deal with the simulation of knowledge transfers, such as Grum and Gronau (2018a), exist. Various forms of simulations are applied for process simulations.
- ⇒ Faced with a great heterogeneity of simulation approaches, no simulation approach considers process and knowledge simulations using neuronal activations.
 - ⇒ Faced with the simulation of 3D ANN compartments and networks, no simulation approach considers process representations within the real world.
- 9) Process optimizations are fixed components in a serious process management. Further, different kinds of process optimizations result in different kind of adjustments within the process model.
- ⇒ Faced with different kinds of process optimization approaches, no single approach realizes process or knowledge optimizations by neuronal learning procedures.
- 10) Various kinds of optimizations are carried out with ANN approaches. For example, approaches that optimize the portfolio management of assets and credit defaults, chemical procedures, and production maintenance can be found.
- ⇒ Faced with a great heterogeneity of ANN approaches, learning is carried out by optimizations. But no learning integrates with the process domain.
- 11) Process modeling and simulation tools are mostly selected on the basis of the fulfillment of criteria. Various criteria catalogs and selection systems can be found in literature. Here, current criteria focus only on the traditional process domain.
- ⇒ Current criteria do not capture the use of neuronal artifacts, such as neuronal techniques and neuronal procedures, at all.
 - ⇒ Current criteria do not capture the use of CoNM-required modeling entities.
 - ⇒ A CoNM-suitable process modeling tool evaluation system is missing.

- 12) ANN tools are mostly selected on the basis of their performance with regard to specific learning problems. While each approach focuses on its own uniqueness, various criteria are presented in literature, but current criteria focus only on the ANN domain.
 - ⇒ All-encompassing multi-aspect criteria catalogs and selection systems for ANN tools are missing.
 - ⇒ Current criteria do not capture the use of business processes, economic simulations, and knowledge management at all.
 - ⇒ Current criteria do not capture the use of CoNM-required modeling entities.
 - ⇒ A CoNM-suitable ANN tool evaluation system is missing.
- 13) Process modeling, simulation, and ANN tools provide a high specialization with regard to their domain. A vast number of hardly-distinguishable tools can be found in the process domain literature. Tools of the ANN domain primary are distinguishable by the concepts provided.
 - ⇒ Faced with a great number of modeling and simulation tools as well as ANN tools, neither do tools of the ANN domain integrate with tools or concepts of the process domain nor do tools of the process domain integrate with tools of concepts of the ANN domain.

Altogether, the thirteen limitations and issues presented characterize the research gap. This can be identified to lie in between the *process domain* and the *ANN domain*, in the intersection of the four relevant topics of *knowledge modeling*, *learning*, *software tooling*, and *process optimizing* (see Fig. 1.1). Since the construction of a CoNM integrates the domains and connects topics, it is the most important and very first step for the specification of the *ANN process domain*. Based on the specification of the *ANN process domain*, a CoNM will be constructed in the following chapters as it intends to address the research gap presented.



Objectives and Methodology

3

The problem analysis and the research gap identified in the previous section stand as the reference point to substantiate the objectives of this contribution. In accordance with Peffers et al. (2007), this is the second step of a design-oriented research proceeding. Objectives for a solution are defined so that the proposed research design (see section 1.3) can be concretized with content. This clarifies the methodological approach of this contribution and the objectives serve as orientation in consecutive sections. Hence, this section is structured as follows:

First, it will be clarified how the CoNM, which was identified by the research gap, is part of a larger and coherent junction of process design, process simulation, process optimization maxims, and learning capabilities of ANN, including the corresponding knowledge representation. With this, the required objectives for that artifact can be specified, and functional as well as non-functional requirements are presented.

Second, methodological additives, which are required for the establishment of a content-based proceeding, are derived from this objective specification.

Third, the concrete proceeding of this contribution is presented by integrating methodological additives. In general, this is a modification of the DSRM of Peffers et al., which is combined with the required methods (mixed method design); additionally, content-based steps are mapped to methodological process steps.

The objective specification and concretized proceeding clarify which terminologies are required as a foundation for this contribution. Hence, terminological basics are presented. With this, the general terminological and methodological framework of this contribution is set up.

Supplementary Information The online version contains supplementary material, such as Appendix A, B, C and D whose references are indicated in the text, available at https://doi.org/10.1007/978-3-658-35999-7_3.

3.1 Analysis of Objectives

The analysis of objectives serves to transfer the conclusions of the problem analysis (see section 2.7) to the specification of the intended solution, which can be done on the basis of a qualitative description of a new artifact (Peffers et al. 2007). Objective analyses are presented as follows:

A first subsection presents the coherent interplay of process design, optimization maxims, knowledge representation, and learning capabilities of ANN. With this, the CoNM terminology is introduced. The second subsection classifies and limits the object of reflection. Hence, in the third subsection, the collection of requirements for an CoNM can be presented.

3.1.1 CoNM Terminology

The terms of *modeling*, *simulation*, and *optimization* are often used in combination. Since they have a different meaning in accordance with their specific domain, the understanding of a model and usage of modeling at the design fields of AI modeling and process modeling has been identified as a research challenge (Grum and Gronau 2018a; Fettke 2020). The following section presents the commonalities and differences to clearly differentiate between these basic terms with regard to the process domain and ANN domain and transfers them to the ANN process domain. Definitions presented here shall close the gap between the process domain and the ANN domain and build on the only approach of Grum and Gronau, which addresses these domains (Grum and Gronau, 2018a). With this, a common CoNM terminology is created that differentiates the terms of *modeling*, *simulation*, and *optimization* with relevant domains thus serving as a conceptual clarification for this contribution.

Process Domain: The term *process modeling* is connected to the creation of abstract illustrations showing only relevant attributes of real-world systems (see sections 2.1– 2.4). This is done with the intention to reduce the complexity with regard to modeling objectives. Here, the knowledge of process engineers is used for the observation, selection, and transfer of a real-world system to homomorphous, time-based, and construction-based mappings (Grum and Gronau 2018b, Rolland, Prakash, and Benjamen 1999) focusing on a sequence-based, plausible visualization (Grum and Gronau 2017). Further, knowledge can be found in reference process designs that have been approved by further institutions and will be considered an inspiration during the modeling (Fettke and Loos 2007). Hence, the experience of process engineers increases through both the consideration of reference processes

knowledge and the doing of process modeling because of human-based learning processes. Once a valid process model has been created, the process model can be tested in real-world or computer-based "play through" runs (Tumay 1995). These are called *process simulations* and require a simulation model (Zhang et al. 2015). This allows the user to characterize different "what-if" scenarios (Rolland, Prakash, and Benjamen 1999) and perform experiments on that model without having to collect additional data (Jung and Hoehe 2015). Since a simulation can give evidence that a process model is not valid (Mahmoud et al. 2009), the two terms process modeling and simulation are closely connected. Here, knowledge lies in the identification and implementation of "what-if" scenarios (Gronau and Grum 2019). The learning capabilities of the simulation engineers guarantee the continuous improvement in the creation of simulation experiments (Schmidt and Taylor 1970, Shannon 1975). One speaks from *process optimization* when processes are adjusted with the intention to optimize them with regard to a certain objective (Teran and Grotewold 1996). Here, the knowledge of process engineers is used for the identification of weaknesses or rather bad patterns within a given process (Niedermann, Radeschütz, and Mitschang 2011) and the creative use of knowledge for their repair (Gronau and Grum 2019). Since process engineers have learning capabilities, the activity of process optimization improves because of a knowledge increase. The result is that the process engineers achieve better optimization results in consecutive runs (Masaaki 1986, Moen and Norman 2006, Shewhart and Deming 1939). The terms process simulation and optimization are closely connected because simulations can lead to estimations of optimization suggestions. When processes are optimized, process models are affected, which is the reason for the close connection between the terms process optimization and modeling.

ANN Domain: The term *neuronal modeling* is connected to the creation of an approximation model by algorithms that focuses on relevant attributes of the real-world data with the intention to reach a certain approximation sufficiency (Hornik, Stinchcombe, and White 1989). It does not follow the intention to reduce the complexity of the real world with regard to modeling objectives. Hence, neuronal models can be understood as a homomorphous, time-based mapping of a real-world system focusing on a sequence-based, sufficient approximated behavior (Hammer 2000, Poggio and Girosi 1990). Visualizations for this are hard to interpret and rare (Yu and Shi 2018). Here, the knowledge of data scientists is used for the identification of required data sources, preparation of training and test data sets, the design of the learning context, the realization of training and test procedures, etc. Often, one speaks of a "black magic" in association with the creation of neuronal networks as though it is an indecipherable black box (Basheer and Hajmeer 2000, Tu 1996,

Yu and Shi 2018): Although the creation of the neuronal model itself is carried out by algorithms following the biological model of the human brain (O'Reilly and Frank 2006), any design decisions mentioned before often are determined through trial-and-error (Kaastra and Boyd 1996, Lämmel and Cleve 2012). Hence, the modeling directly depends on the knowledge and experience of the corresponding data scientist. Since only basic network architectures and algorithms are known and the construction of more sophisticated ANNs is an emerging research domain (Yu and Shi 2018), modeling maxims are rare. As neuronal networks are realized, those can be tested in computer-based “play through” runs to check if they generalize correctly and establish a realistic behavior (Brette et al. 2007, Sarma et al. 2018). Mostly, the whole dataset is split into two parts. While the first part is used as a set during the training, the second part is used as a test set during *neuronal simulations* (Cawley and Talbot 2003, Dor and Zhou 2007, Mitchell 1997, Wang et al. 2020). Those simulations allow the user to analyze the network within the modeled learning context and in different “what-if” scenarios without having to collect additional data. Improvements of ANN refer to changes within the ANN that lead to a higher approximation accuracy, learning speed, and capability of correct generalizations of neuronal networks (Joost and Schiffmann 1998, Yu and Shi 2018). Further, improvements within the real world can be derived through the use of neuronal networks; for instance, ANN outputs change that represent real-world parameters. Examples can be found from the modification of chemical reactions (James, Lam, and Li 2011), process conditions (Shabani and Mazahery 2012, Shen, Wang, and Li 2007), or the manufacturing process of laser welding (Casalino et al. 2016). From an optimization perspective, one can speak from a *neuronal optimization*, because these are based on data and incremental changes (Chambers and Mount-Campbell 2002b).

ANN Process Domain: Since the understanding of a model and the usage of modeling at the design fields of AI modeling and process modeling has been identified as a research challenge (Grum and Gronau 2018a; Fettke 2020), the following intends to close the gap between the process domain and the ANN domain. A CoNM terminology is established by the intersection of the individual terminologies of the process and the ANN domain so that the ANN process domain manifests (see Fig. 3.1). Since this contribution aims to start discussions about the ANN process domain, the ANN process domain must be considered hypothetical as long as proof-of-concepts are missing. It has guided the construction of the CoNM as the design principle.

The ANN process domain is based on the following definitions: a *Neuronal Process Modeling* (NPM) is referred to as the modeling of processes at a neuronal level with a common process modeling language, the reinterpretation of the

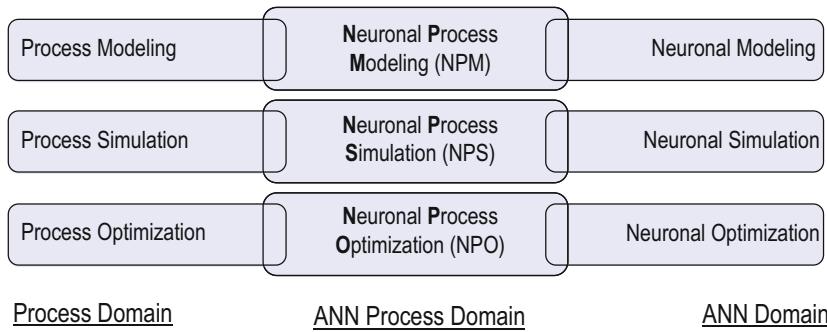


Figure 3.1 The CoNM Terminology as Binding Part at the ANN Process Domain following Grum and Gronau (2018a)

common process modeling based on that understanding as well as their difference quantity (Gronau and Grum 2017, Grum and Gronau 2018a). The *Neuronal Process Simulation* (NPS) refers to the process simulation of common process models considering ANNs as knowledge models of process participants (persons and machines), the simulation of common process models reinterpreted as deep neuronal network, the simulation of neuronal processes reinterpreted as organizational processes, and their difference quantity (Gronau and Grum 2017, Grum and Gronau 2018a). The *Neuronal Process Optimization* (NPO) is referred to as, first, common process optimization techniques that are realized on a neuronal level (Chambers and Mount-Campbell 2002a, Chambers and Mount-Campbell 2002b, Davis and Gasperi 1997, Gronau and Grum 2017, Grum and Gronau 2018a) - examples for the same can be found at the double-loop learning on a neuronal level - second, novel forms of process optimizations that can be realized because of the learning capabilities of ANN in the domain of common process models, and, third, their difference quantity (Gronau and Grum 2017, Grum and Gronau 2018a).

Within those definitions, a *neuronal activity* refers to the computational activation of a single neuron and a *neuronal task* to a computational activation of a set of neurons, which includes the integration of the activation of former neurons and a corresponding designation as it is usual in the process domain (Grum 2020). Hence, a neuronal task and a neuronal activity is affected in all, NPM, NPS, and NPO.

3.1.2 Classification and Limitation of the Object of Reflection

To show which tasks shall be realized with the help of the CoNM, the object of reflection is classified within its research context. Then, limits are identified, so that detailed requirements can be derived in the next section.

Classification Approach: The embedding of the CoNM in its research context can be seen in Fig. 3.2. This will serve as a conceptual reference frame for the construction of the CoNM. Within the figure, the CoNM can be found as a systematic collection of domain-specific artifacts within the interplay of theory, neighboring domains, and reality. Their relations and dependencies are explained in the following section.

In accordance with the CoNM terminology and definitions of Fig. 3.1, the CoNM is situated between the process domain and the ANN domain. Following the methodology definition of Weber, a domain-specific methodology is a system of methodological artifacts with the objective to recognize and affect the reality with regard to the object of reflection (Weber 2015, p. 67). While each domain-specific methodology obtains its purpose from its corresponding theory and provides its own terminology, the CoNM is affected by both the process domain and the ANN domain. Each domain-specific methodology provides specific artifacts. Those include methods, tools, algorithms, techniques, hardware, software, etc. Since those are applied in reality and get impulses through the application, a user-centric perspective is essential. As deep network-based artifacts refer to great numbers of layers within neuronal networks (hundreds of layers are usual) and the difficulty to gain an overview increases the deeper networks are, a focus of this contribution lays on visualizations for the user.

Since the theory can be affected through methodological changes and a consequent mapping of the reality, the theoretical foundation is able to always be up-to-date. Since the CoNM is affected by both the process domain and the ANN domain, each domain can provide domain-specific artifacts for the CoNM. Hence, the process domain serves as a superset from the modeling side, and the ANN domain serves as a superset from the neuronal technique side. In contrast, the CoNM provides ANN process domain specific artifacts that are assumed to expand the methodology of the process domain and the ANN domain. Hence, both domains require the integration of the CoNM as the enabling concept as well. Since the CoNM is created with the intention to affect the reality, the CoNM is assumed not to be a loose collection of artifacts but a systematic collection tightened by a procedure model, which shall ensure a common use. With this, the CoNM can be applied to affect the

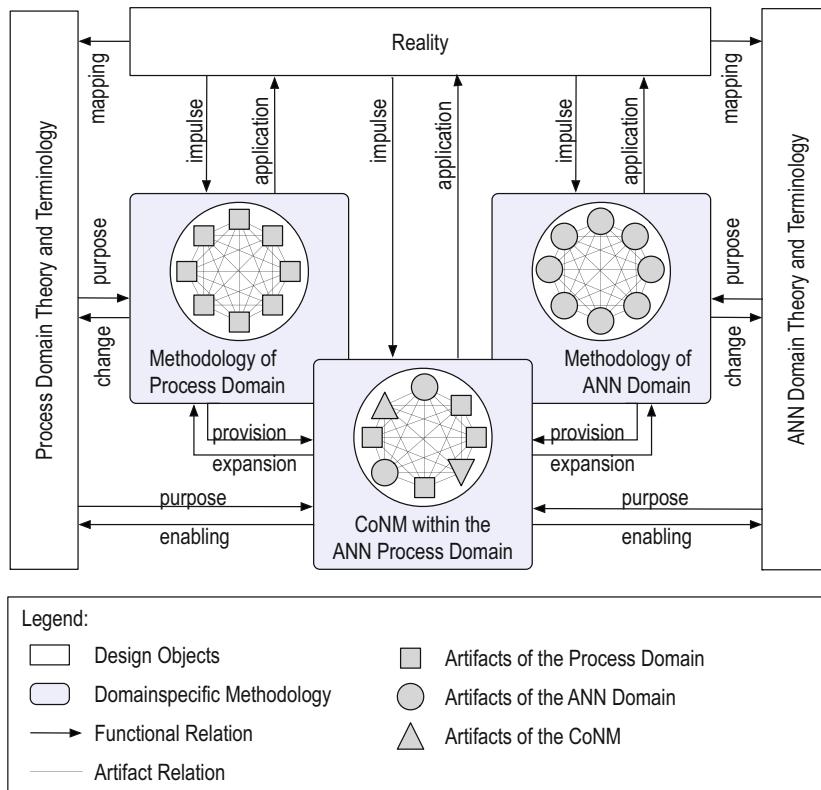


Figure 3.2 Conceptual Reference Frame of the CoNM

reality such that the impulses affect all kinds of domain-specific methodologies and domain-specific mappings to theory and terminology.

Limitations: Although the process domain and ANN provide numerous artifacts that is of great potential for the CoNM, this contribution does not try to transfer all the available artifacts to the ANN process domain. It focuses on the selection and transfer of a minimum of artifacts with the intention to proof presented concepts. Besides, those examples serve as a framework for the selection and transfer of further artifacts. Further, this contribution does not try to transfer all available concepts with

help of the CoNM from the process domain to the ANN domain and vice versa. Although this would provide a great potential, the focus is on the proof of concepts. As new artifacts are created within the ANN process domain, this contribution does not try to map those to the process domain and to the ANN domain. In contrast to the theory methodology mapping of the process domain and the ANN domain, this contribution does not try to provide an individual ANN process domain theory.

Interim Conclusion: As a result of this classification and limitation, the following elements can be characterized as essential for the CoNM:

1. Process domain-specific (modeling side), ANN domain-specific (neuronal technique side), and new ANN process domain-specific artifacts are the foundation of the CoNM.
2. Single artifacts are connected by a proceeding to guarantee the consistent use of the CoNM.
3. The use of the CoNM is based on the integration of all parts of combined artifacts, which considers hardware, software, user-friendly visualization, etc.
4. The application of the CoNM shall prove the concept using scenarios.

The presented classification and limits serve as foundation for the requirements.

3.1.3 Requirements

Capabilities that an artifact has to provide and constraints that have to be fulfilled by the artifact to satisfy as solution for the identified problem are referred to as requirements in the following section. The presented section presents requirements that have been derived from objectives of the CoNM (see section 1.2). As section 8.2.2 will clarify, these have been discussed with experts from the domain of knowledge management, process modeling and optimization, simulation, and AI and have been optimized over the course of three years. Hence, the following presents final versions only.

Process Domain Side: Assume to have a given process model and process data that can be used to create a neuronal network representing the knowledge and behavior of a process participant. Then, one can consider this neuronal network as a process participant's knowledge model within the models and simulations of that initial process model. Here, the following requirements coming from a neuronal side have

to be considered on the *Process Domain* side (subsidiarily to Gronau and Grum 2017; Grum and Gronau 2018a):

1. Neuronal knowledge models have to be integrated within process models so that the approximation capabilities of ANN can affect process models.
2. The very same neuronal knowledge model must be able to be integrated several times within a process model. This follows the metaphor of having referenced ANN approximating process participants that face different business processes within a work day. Hence, activations result as the preceding tasks must not be deleted, and the neuronal network remembers its process history.
3. Neuronal knowledge models have to be integrated within process simulations since the simulation of knowledge carriers is based on their knowledge.
4. Modeled environmental factors (material such as non-material objects) have to be integrated into prior mentioned knowledge models so that a common level of interpretation is enabled.
5. Outcomes (materialized such as non-materialized) of prior-mentioned knowledge models have to be considered in the process models and within process simulations so that the effect of knowledge transfers can be observed.

ANN Domain Side: Assume to have a given ANN model that shall be mapped to a process model representing the logic and behavior of the process observed. Then, one can consider this neuronal network within the models and simulations of that initial process model. Here, the following requirements coming from the process side have to be considered on the *ANN Domain* side (subsidiarily to Gronau and Grum 2017; Grum and Gronau 2018a):

6. Neuronal tasks have to be considered while neurons follow biological models. This includes the neuron's operational behavior and its learning processes.
7. Parallel neuronal tasks have to be considered within neuronal networks. Here, process models provide tasks and activities that are realized simultaneously.
8. Time-dependent neuronal task behaviors have to be considered within neuronal networks. Here, process models provide tasks and activities that evolve over time. Mostly, this is characterized in the simulation scenario.
9. Sequential neuronal tasks have to be considered within neuronal networks. Here, process models provide tasks and activities that are realized one after the other.
10. A control flow as it is realized with the help of Boolean operators has to be considered within neuronal networks. Here, process models provide junctions of task and activity sequences and decisions are modeled.

11. Different levels of neuronal task abstractions have to be considered in the neuronal process modeling and simulation. Here, process models provide models of various levels of abstraction and granularity.
12. Sensory information and knowledge flows have to be considered within the modeled neuronal network.
13. Actuator information and knowledge have to be considered the outcomes of neuronal networks.

ANN Process Domain Side: Assume to have created a joint understanding of processes and neuronal network based on ANN. Then, one can consider operations of the one model on the other model and vice versa. Here, the following requirements coming from a modeling side have to be considered from the process and neuronal sides.

14. When faced with various kinds of experts and domains each considering an individual terminology, it is required to set up a joint framework for wordings and contexts.
15. Complex ANN structures demand for the human-friendly visualization so that model constructors with different competences are enabled to cooperatively work on one system. For example, one can find management experts with a BP perspective on the ANN system, NI experts with a biological perspective on the ANN system, KM experts addressing the knowledge management within the ANN system, simulation experts focusing on the ANN system use in simulations, as well as ANN and ML experts focusing on the ANN system performance, etc. This further includes human as well as machine-based model constructors. Here, it is required to identify one joint language that can serve as a communication platform for all.
16. As process models are commonly visualized in 2D and optimized for printouts, a visualization of neuronal process models, which shall follow real biological models, may not be limited by those two dimensions. A visualization in 3D is attractive for the following reasons: the brain exhibits networks with inhomogeneous connectivity probabilities (Lübke et al. 2003), spatial clustering, and reinforced multicell motifs (Song et al. 2005, Sporns and Kotter 2004). All of them can be traced back to local connections and the presence of local circuits (Yoshimura and Callaway 2005). They are essential for the computation (Pouille and Scanziani 2004). Since both the physical position of neurons on the one hand and the thickness and physical routing of connections on the other are intended to follow the biologic model of the human brain, a visualization in 3D is required.

17. As neuronal process models can transfer the behavior of the human brain to further contexts, such as the economy, production, etc., a projection in the intended working space is required.
18. As ANN systems non-transparently approximate functional relations, different kinds of ANN systems can be identified that all have an individual demand for complex cognitive structures: for instance, one can find systems issuing the behavior of a biological NN. Further kinds issue the approximation of a specific task, while other kinds address the knowledge generation of knowledge carriers. Hence, their approximation capability is required to follow an architectural design that enables dealing with different complex kinds of cognitive objects. This can be oriented to layers, cortical maps, or topographic structures following contemporary NI insights (see section 2.5.1.5).
19. As the modification of ANN models is circumstantial and the modeling language established shall be used to a) support the ANN construction, b) enable an insight in the current working of ANN, and c) support the reuse of valuable ANN models, the modeling language demands a technical substantiation.
20. In organizational contexts as well as in future contexts, ANN systems are enabled by numerous input sources and heterogeneous data stream origins. Particularly, the organizational environments hold numerous kinds of data sources such as hardware systems, software systems, and databases. All of these need to be considered jointly as potential input and target sources for ANN systems so that an enterprise architecture can consider the systematic and sustainable use and reuse of data.
21. As different kinds of experts are required to cooperatively work on one joint ANN system, a measure-based management model is required, which enables the cooperative model modification of various kinds of experts (human as well as machine-based managers). This considers all—the measures affecting the domain of knowledge management, the measures affecting the neuronal operation, and the measures affecting all.
22. As models constructed can be considered for different purposes, such as the NPM, NPS, and NPO (see section 3.1.1), they need to be assumed to have a life-span. Further, as the CoNM needs to be considered as a novelty that needs to be enrolled to companies, a project-oriented consideration needs to be established that enables the phase-specific project realization and presents quality gates per phase. Hence, a procedure that enables the controlled, quality-assured, cooperative model construction of various kinds of experts is required.
23. ANN systems can function as an organizational control center: they can perceive the organizational environment first—an example of this can be found in the data stream input from production systems. They can affect the organiza-

- tional environment second—an example of this can be found in the instruction realization with the aid of workflow systems.
- 24. As one is faced with a great collection of tools, each realizing an individual specialization for process modeling (section 2.1), knowledge modeling (section 2.2), simulation modeling (section 2.3), and process optimization (section 2.4), and the tools differentiate in their need for CoNM implementation effort, a systematic evaluation system is required. This must consider detailed and concrete criteria addressing CoNM-relevant capabilities so that the tools having a best requirement fulfillment can be selected.
 - 25. As it is not clear how to identify and visualize the knowledge generation of ANN structures, a procedure for the systematic extraction of knowledge from ANN systems is required. This must enable the compilation of a neuronal knowledge base that must be integrated into the organizational knowledge base.
 - 26. As the CoNM intends to consider tempo-spatial relations and operates in real-world dimensions, a training procedure that unlocks capacities at a certain location within the ANN is required.
 - 27. As the CoNM intends to operate in a hierarchical knowledge construction setting, a training procedure that enables the respecting of specific system levels when necessary is required.

Implementation of Hardware and Infrastructure: The realization of the CoNM is connected to the requirements referring to computing units, as follows:

- 28. Since the training processes of neuronal networks require heavy computer-intensive calculations, a central computing structure for training runs is required.
- 29. As the test of trained networks and use in process simulations (especially systematic simulation runs and sensitivity analyses) cause computer-intensive calculations, a central computing structure is required for simulation runs.
- 30. Since the training and test processes of neuronal networks need huge amounts of data, a central memory structure is required. Focusing on the integration of production systems and the use of real-time data streams make the integration of analytic platforms attractive.
- 31. The construction of neuronal networks can be realized locally on ThinClients, laptops, desktop computers, etc., since this is not computationally intensive but affordable for human model creators.
- 32. Coordination efforts of various hardware units shall be realized in the background and comfortably provided via the Internet and WLAN structures.

Implementation of Software Prototypes: The use of the CoNM shall be supported by software so that an easy and consistent access to the CoNM can be guaranteed:

33. The initialization of neuronal networks shall be carried out comfortably via drag and drop mechanisms.
34. The integration of modeling tools and programming libraries shall be realized in the background system. This includes the automatic network structure creation at the base of neuronal process models.
35. The integration of simulation processes and programming libraries shall be realized in the background system. This includes the start, stop, and further control elements of neuronal process simulations.
36. The integration of optimization processes and programming libraries shall be realized in the background system. This includes the start, stop, and further control elements of neuronal process optimizations.

Demonstration and Evaluation: For an evaluation of the concept, the functioning of the CoNM has to be proven based on neuronal experiments. Each provides an individual focal point and is presented as a proper scenario. Regarding the scenario creation, the following requirements have been identified (subsidiarily to Grum and Gronau 2017):

37. Illustrative examples shall visualize basic ideas of the CoNM. Although this limits the practicality of the insights generated, it greatly supports the comprehensibility of the CoNM modeling concept.
38. According to different approaches to build model hierarchies, these refer to a decomposition of super-components (top-down modeling) or a composition of sub-components (bottom-up modeling), or both (hybrid modeling), which are well known as modeling characteristics (section 2.1.1.4) and manifested by the NMDL's atomic taxonomy views (section 4.6.4); each needs to be considered in one demonstration at least.
39. Since ANN can approximate the behavior of any kind of system, different kinds of interacting knowledge carriers need to be considered in the demonstration.
40. According to the CoNM proceeding to be specified, phases need to be demonstrated. If established in experiments, a focus needs to consider at least one phase.
41. According to the modeling language to be designed, which integrates the *Process Domain* and the *ANN Domain*, the interplay of different kinds of perspectives, views, and items is required to be demonstrated.

42. According to the CoNM terminology established in section 3.1.1, experiments shall present examples for the NPM, NPS, and NPO. This focuses on the functional proof of the CoNM.
43. As the modeling can be realized from both humans as well as machines, each kind of model constructor needs to be demonstrated.
44. According to the measure-based management model to be designed, its dimensions need to be demonstrated at least once.
45. According to the specification of different kinds of ANN systems, these refer to (1) network systems, (2) process systems, (3) activity systems, and (4) activation systems; these kinds of ANN systems need to be demonstrated.
46. The novel CoNM algorithms need to be demonstrated. This for instance refers to the algorithm called SEKO (section 4.7), which is in regard with the identification of tacit and explicit forms of knowledge. Further, this includes new learning principles (section 4.8) addressing different forms of ML algorithms within the *ANN Process Domain*.

Each requirement of these categories has been relevant during the construction of the CoNM and serves as an input for the following sections.

3.2 Methodological Additives

The collection of methodological additives serves as the foundation for the establishment of a concretized proceeding of this contribution. On the basis of their realization, isolated artifacts are carried out and demonstrated. Individual conclusions are finally integrated, as is explained in section 3.3. Methodological additives are presented as follows:

The first subsection selects a methodology for the realization of a systematic literature research. The second subsection presents a methodology for the realization of design-oriented research following Peffers et al. The third subsection presents a methodology for the creation of neuronal networks following Lämmel and Cleve. The fourth subsection presents a methodology for the realization of empirical research following Diekmann et al. The fifth subsection presents a methodology for the integration of individual methodologies following a mixed methods design.

3.2.1 Method for the Systematic Literature Review

Before a contribution creates new artifacts, its place in the current research needs to be identified (Webster and Watson 2002). In accordance with the research definition presented by Leedy and Ormrod, artifact creation is connected with the endeavor to enhance the understanding of a phenomenon and the communication of new insights to a large scientific community (Leedy and Ormrod 2005). Hence, an essential first step for any research project is to identify the current status of knowledge in the corresponding research field—the so-called *body of knowledge* (Iivari, Hirschheim, and Klein 2004).

Following Levy and Ellis, a systematic literature review (SLR) identifies the body of knowledge and supports the communication process in the following ways (Levy and Ellis 2006):

- The SLR helps the researcher to understand the existing body of knowledge (What is already known?).
- The SLR helps the researcher to identify where new research is needed (What needs to be known?).
- The SLR provides a theoretical foundation for the proposed research project (What knowledge should the research be based on?).
- The SLR substantiates the presence of the research problem (Where is a research gap?).
- The SLR justifies the proposed research project as contributing something new to the current body of knowledge (How can artifacts enhance knowledge?).
- The SLR frames the search process with valid methodologies, approaches, goals, and research questions (How can the process be structured?).

In the sense of an evidence-based research, the SLR applies best-quality scientific studies on a specific topic or research question and helps build a synthesis of the current understanding of a phenomenon so that appropriate components can be used for the solution of a problem (Kitchenham et al. 2009). In contradiction to the ordinary literature reviews, an SLR is distinguished by the following (Budgen and Brereton 2006):

- A *review protocol* specifies the research questions and methods that will be employed.
- A defined *search strategy* is employed for identifying as much of relevant literature as possible.

- A *search strategy documentation* is provided, enabling any reader to assess the rigorousness and completeness.
- A collection of *inclusion* and *exclusion criteria* is specified to characterize potential contributions.
- A collection of *information* is specified, which is used for evaluation purposes.

These systematization points make the SLR the essential step for the identification of the current status of knowledge in the given research field because its accomplishment becomes transparent and reproducible.

As Fig. 3.3 shows, the systematization of a literature review is realized by representatives differently. While some representatives provide detailed checklists (Liberati et al. 2009), others provide detailed review processes (Kitchenham et al. 2009) or abstract process steps with extensive process descriptions (Levy and Ellis 2006). Although individual literature review steps are denominated differently, in general, they demand similar activities. Therefore, the following section synthesizes approaches and establishes a methodological specification for this contribution.

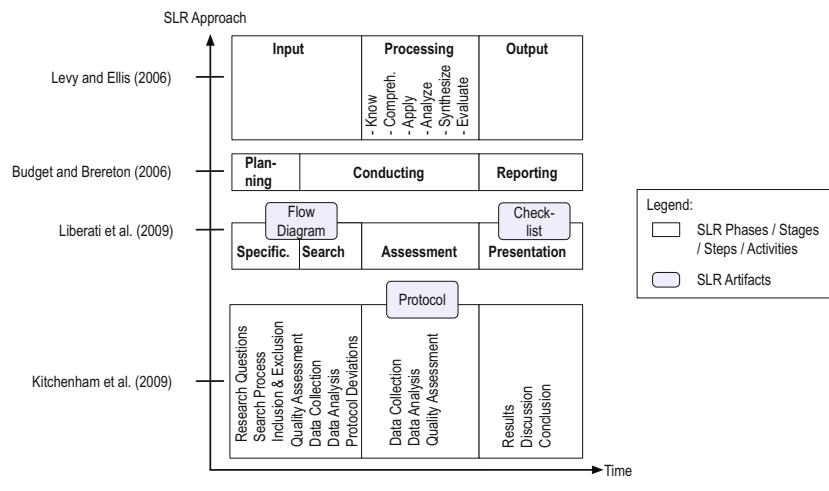


Figure 3.3 Methodological Approaches for the Realization of a Systematic Literature Review

Stage 1—Input: The first SLR stage deals with the acquisition of the *complete census of literature* of a domain (Webster and Watson 2002), the identification of

relevant literature out of this complete census, which refers to a research objective (Kitchenham et al. 2009), and the assurance of its high quality (Levy and Ellis 2006). Denominated as input for consecutive stages, this literature serves as the body of knowledge.

At the very beginning, the SLR needs to be planned (Budgen and Brereton 2006, Kitchenham et al. 2009, Levy and Ellis 2006, Liberati et al. 2009). This demands the operationalization of various dimensions through which the search is then carried out. The following list presents a collection of dimensions, for which the representatives ask for:

- *SLR method*: The methodological approach of the SLR must be specified.
- *SLR objectives*: A reasoning must be provided as to why the SLR was carried out. This can be oriented to the general SLR objectives (Levy and Ellis 2006), but must consider the greater picture of the intended research project.
- *Research questions*: Research questions addressed by the SLR need to be stated explicitly (Kitchenham et al. 2009). Since those will be answered in the course of the SLR, they focus on the existing literature and are different from the main research questions.
- *Search process*: A description of the search process must be given (Kitchenham et al. 2009). This includes the selection of journals, conferences, searched databases and, so far, as several researchers are part of this process, how the collaboration is organized.
- *Search terms*: A description must be provided that presents a selection of *search words* or *phrases* used for the querying of quality scholarly databases (Levy and Ellis 2006). This includes a reasoning for the selection and the construction of search term combinations in the form of Boolean search commands.
- *Keyword searching*: A description must be provided that specifies the categories against which a keyword search has been realized (Levy and Ellis 2006). Examples can be the following: title, abstract, the documents' keywords or full text. This includes backward and forward keyword searches (Webster and Watson 2002) as follows: First, a *backward references search*, which considers references of the articles identified in the keyword search. The levels of backward references under consideration must be specified. Second, a *backward authors search*, which refers to reviewing articles of authors published prior to the article. Third, a *previously used keywords search*, which considers keywords of articles identified in the keyword search as additional search terms. Fourth, a *forward references search*, which refers to the consideration of articles that cite the articles identified in the keyword search. Fifth, a *forward authors search*, which refers to reviewing articles of authors being published after the article.

- *Inclusion & exclusion criteria:* A collection of criteria must be described that records must satisfy to be included in or excluded from the census of relevant literature (Kitchenham et al. 2009). Example criteria for both kinds can be the following: topics, publication dates, kinds of publications, the availability of research questions, methodological structured search processes, concept uses, etc.
- *Quality assessment:* Questions that serve to support the evaluation of literature (Kitchenham et al. 2009) and the cognitively demanding activities of the second stage need to be specified (Bloom et al. 1956).
- *Data extraction:* A tabular overview of literature search results needs to be specified that supports the extraction of record data—the so-called protocol (Kitchenham et al. 2009). As the protocol was already published, deviations from it must be explained.
- *Searching stop:* The rule is to be specified when the SLR is final. By default, the search process should continuously be done during the course of the research project (Levy and Ellis 2006). Alternatively, it can be stopped when no new citations are discovered (Webster and Watson 2002) or the researcher discovers that new articles only introduce familiar arguments, methodologies, findings, authors, and studies (Leedy and Ormrod 2005).

The collection of points presented will serve to systematically plan the SLR in this contribution. The PRISMA statement separates the operationalization of the SLR objectives and the systematic search process (Liberati et al. 2009). While the first can be oriented to a detailed *checklist* being considered above, the latter follows four steps of a *flow diagram*:

- *Identification:* The first stage focuses on the search for records. Outcomes are the number of records identified through database searching and through other sources.
- *Screening:* The second stage focuses on the filtering of records identified in the first stage. Here, duplicates are removed, and records are screened on the basis of title, abstract, keywords, etc. Outcomes are the number of records after each filtering step.
- *Eligibility:* The third stage focuses on the confirmation of the eligibility for the SLR. Here, records are assessed on the basis of the full text. Outcomes are the number of records after each filtering step.
- *Included:* The fourth stage focuses on the differentiation of qualitative and quantitative contributions. Ideally, the collection of quantitative contributions is big

enough to build a statistically secured synthesis. Outcomes are the number of qualitative and quantitative records.

The steps of this flow diagram will serve to systematically conduct the search process of the SLR in this contribution.

Stage 2—Processing: Based on the identification of relevant literature, data contained in the sources must be processed into information suited for new research to build up (Bem 1995). Unfortunately, the PRISMA statement only demands concrete outcomes on the basis of a literature assessment and does not explain how the assessment is carried out. Since an extensive protocol is provided (Liberati et al. 2009), assessment tasks are demanded implicitly. Similarly, Budget and Breton only mention the performance of a literature review and implicitly demand processing activities. The latter are not specified in greater detail. However, Levy and Ellis consider a series of steps to achieve the *cognitive capability* required to transform raw data of numerous literature sources into an effective SLR (Levy and Ellis 2006). Steps are based on the *Bloom Taxonomy* (Bloom et al. 1956) and demand the following more cognitively demanding activities:

- *Know the literature:* The researcher must demonstrate they have read the article and can extract meaningful information from it. Commonly, the knowledge level is demonstrated by activities such as listing, defining, describing, and identifying.
- *Comprehend the literature:* The researcher must demonstrate they can understand the meaning and significance of information being reported and can apply cognitive levels addressed in manuscripts. Commonly, the comprehension level is demonstrated by activities such as summarizing, differentiating, interpreting, and contrasting and includes *theories, constructs, variables, models, and frameworks*.
- *Apply the literature:* The researcher must demonstrate they can apply their comprehension level to identify relevant concepts. A *concept-centric approach* proves this since sources need to be categorized correctly by the researcher (Webster and Watson 2002). Here, the researcher must relate articles to a collection of concepts. Commonly, the application level is demonstrated by activities such as demonstrating, illustrating, solving, relating, and classifying.
- *Analyze the literature:* The researcher must demonstrate they can apply their comprehension level and analyze relevant concepts in regard to units of analysis. A *concept matrix augmented with units of analysis* proves this since sources need to be categorized correctly in regard to various analysis dimensions (Webster and

Watson 2002). Commonly, the application level is demonstrated by activities such as separating, connecting, comparing, selecting, and explaining.

- *Synthesize the literature:* The researcher must demonstrate they can assemble the literature for a given concept into a whole that exceeds the sum of its parts. Here, a harmonized big picture is created that deals with conflicting concepts at the same time. Commonly, the knowledge level is demonstrated by activities such as combining, integrating, modifying, rearranging, designing, composing, and generalizing.
- *Evaluate the literature* The researcher must demonstrate they can clearly distinguish between opinions, theories, and empirically established facts. Commonly, the knowledge level is demonstrated by activities such as assessing, deciding, recommending, selecting, judging, explaining, discriminating, supporting, and concluding.

In addition, Kitchenham et al. provide more abstract activities and relate the processing directly to the concrete design of data structures required for the collection, analysis, and quality evaluation of relevant sources. A *protocol* needs to be specified for this (Kitchenham et al. 2009).

The collection of these activities can serve to systematically process the SLR in this contribution.

Stage 3—Output: Based on the processing of relevant literature of the previous stage, the third stage focuses on the communication of SLR results (Budgen and Brereton 2006, Kitchenham et al. 2009, Levy and Ellis 2006, Liberati et al. 2009). A communication should provide the reader with what the researcher did during all SLR stages. As the SLR is part of academic writing, the following is required (Hart 1998):

1. The SLR's writing must be clear.
2. The SLR must have a logical structure.
3. The SLR must show the acquisition of capabilities at the appropriate level.

Hart's SLR requirements can be realized as follows: First, a clear writing is realized when the argumentation follows common argumentation theories (Levy and Ellis 2006). Second, a logical structure is realized when the writing is structured by phases and the proper argumentation (Levy and Ellis 2006) by checklists (Liberati et al. 2009), or protocols (Kitchenham et al. 2009). Third, the acquisition of relevant capabilities is shown when the writing considers Bloom's cognitive capabilities

(Levy and Ellis 2006). The collection of these concepts will serve to systematically communicate the SLR in this contribution.

Iterative Proceeding: Since the realization of an SLR is a creative process and insights of consecutive stages can be reflected in regard to outcomes of former stages, these stages are not carried out in a strict sequence. An iterative process can lead to robust research results and supports the identification of the complete census of relevant and high-quality literature. Hence, visualizing all, Fig. 3.4 presents the methodological foundation for the SLR of this contribution.

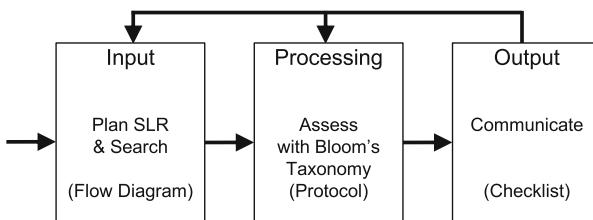


Figure 3.4 Methodological Approach for the Realization of a Systematic Literature Review

Intending to create a CoNM and to build on the existing concepts, the SLR method presented here serves as a reference point to identify a theoretic foundation for the creation of sub-artifacts and the main artifact of the CoNM. Further, it helps identify the research gap that the CoNM addresses.

3.2.2 Design Science Research Methodology

Being a design science research method, the Design Science Research Methodology (DSRM) of Peffers et al. is an approach that guides researchers through design science with the help of a process model (DSRMP) and a mental model (DSRMM) with the aim of efficiently producing and communicating design science research. The DSRM of Peffers can be used to structure the research since it is based consistently on design science literature and leads to sufficiently complete and robust research (Peffers et al. 2006, p. 102). The mental model can be used for the evaluation of theory testings and for presenting design science research. Hence, the mental

model can aid the derivation of components that one can expect from a publication (Peffers et al. 2007, p. 2). So, it is *a priori* clear which parts the research output must contain. The DSRM is described in the following section and can be seen in Fig. 3.5.

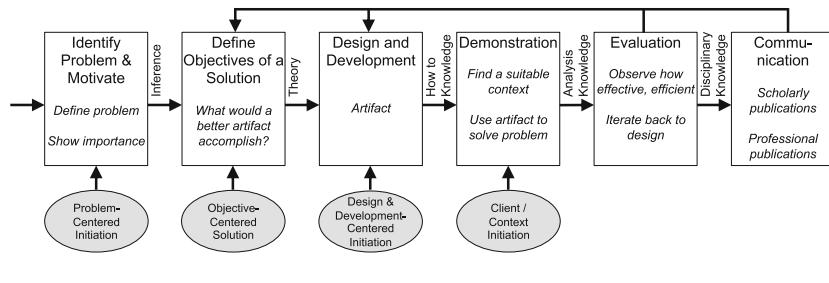


Figure 3.5 The Design Science Research Methodology Process Model (Peffers et al. 2007)

Within the figure, six process steps are visualized. Although they are not meant to be processed sequentially (Peffers et al. 2007, p. 14), they are connected by arrows pointing to the right, which is called the *nominal process sequence*. In the following section, the six process steps are described in accordance to Peffers et al. (2007, p. 12–14):

Activity 1—Problem Identification & Motivation: A research problem is defined and the value of a solution is justified to motivate the audience to pursue the solution and accept the results. For this, required knowledge resources are taken into account. This helps to understand the reasoning of the researcher. Since problem definitions can be quite complex, reducing them to the atomic components of the problem may be useful.

Activity 2—Define the Objectives of a Solution: Objectives are inferred rationally from the previously defined problem specification. These are referred to as requirements. For this, required knowledge resources are taken into account besides current solutions, which help to understand the reasoning of the researcher.

Activity 3—Design and Development: Considering the previously defined objectives, the artifact is designed and developed. Its foundation can be any existing

artifact in which a research contribution is embedded. To support the progress of designing and developing, theoretic knowledge resources are used.

Activity 4—Demonstration: The previously built artifact is used to solve one or more instances of a problem. Computer simulations, empirical applications, theoretical proofs, and real experiments can serve as demonstration. Hence, required knowledge resources (how-to knowledge) focus on the demonstration purpose.

Activity 5—Evaluation: Based on the impressions of the demonstration, the artifact is assessed in regard to how well it supports the solution of the problem. Depending on the problem's nature, relevant metrics, analysis techniques, and various kinds of performance measures, any evidence or proof can serve the evaluation purpose. Corresponding knowledge resources are used (analysis knowledge) for this process. If the evaluation shows an appropriate solution, the research continues with activity six. If the solution is not appropriate, a decision is made to revert to activity two or three depending on necessary adjustments.

Activity 6—Communication: In order to diffuse new insights, previously worked out research components are communicated to other researchers and relevant audiences. Here, disciplinary knowledge is considered beside cultural knowledge in order to find an appropriate communication style.

Inspired by those activities, a research process can be initiated through four possible entry points (Peffers et al. 2007, p. 14–15). Those are visualized in Fig. 3.5 with the help of ellipses:

Entry Point 1—Problem-Centered Approach: If a researcher observes a certain problem or wants to build up on outlined problems, they are following a problem-centered approach and realize the nominal process sequence. Hence, they start with the first and end with the last activity of the DSRMP.

Entry Point 2—Objective-Centered Solution: The need to start with the second activity would result from an industry's or research's need to develop an artifact based on a given collection of objectives. Nevertheless, a problem definition and motivation has to be found before the nominal process sequence can be realized.

Entry Point 3—Design and Development-Centered Approach: Triggered by the existence of an artifact that has not been used as a solution within a certain problem domain, the third activity can serve as an entry point. Initial activities have to be realized before the nominal process sequence can be realized.

Entry Point 4—Client/Context-Centered Approach: Having observed the behavior of a working solution within a practicable scenario, the fourth activity can serve as an entry point to creating a design science solution. Initial activities have to be realized before the nominal process sequence can be realized.

Since the realization of those activities is not carried out in a strict sequence and both the evaluation and the communication can result in the renewed consideration of the second and third activity, an iterative process can lead to robust research results. Hence, the DSRM is an attractive choice for the creation of the CoNM artifact. Since the creation of the CoNM artifact and its sub-artifacts is connected to development and implementation, the method presented here serves as the methodical reference point in this contribution.

3.2.3 Method for the Development of Neuronal Networks

Since a common success strategy for the development of neuronal networks is not available yet, the development of neuronal networks can only be realized on the basis of experiments (Lämmel and Cleve 2012, p. 198). This is carried out in cyclic phases, as can be seen in Fig. 3.6.

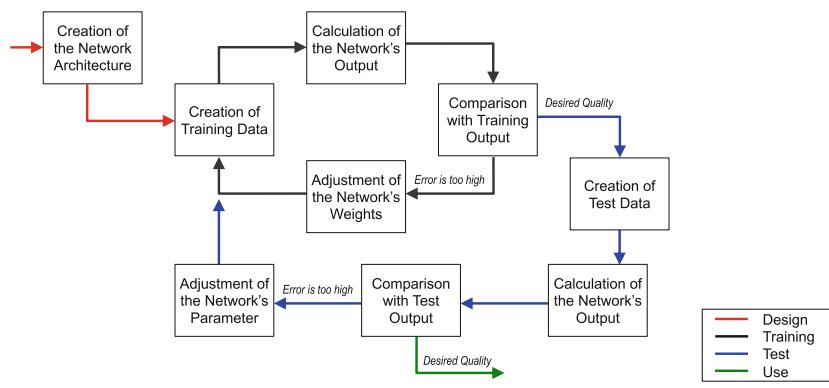


Figure 3.6 Methodological Approach for the Development of Neuronal Networks following Lämmel and Cleve (2012, p. 199)

Within the figure, nine process steps are visualized, which are connected by colored arrows indicating the *nominal process sequence*. Nine process steps are

described within the following four phases in accordance to Lämmel and Cleve (2012, p. 199):

Design Phase: At the beginning, a network architecture is selected in dependence on the learning problem the network shall deal with. Even the application context of the neuronal network can give evidence for the selection of a certain architecture. Quality criteria for the training and test cycles are defined, and the learning parameters and learning algorithms are specified. A collection of network architectures can be found in section 2.5.8.

Training Phase: For the successful development of a neuronal network, a sufficient number of training data is required. This is prepared for the training process in a first step (*Creation of Training Data*). The training is realized as it was defined in section 2.5.6: The training data is used for the neuronal activation in a second step (*Calculation of the Network's Output*). The third step (*Comparison with Training Output*) establishes the training error \underline{E}^T as a difference between the network's current output and the intended training output (see Eq. 2.45 and Eq. 2.47). If the training error is not in accordance with the definition of the desired training quality, the network's weights are adjusted in a fourth step (*Adjustment of the Network's Weights*) following Eq. 2.44. Then, the training phase is repeated until the training error is appropriate for the definition of the desired training quality.

Test Phase: In order to assure a successful generalization, a neuronal network is tested during the test phase. Here, a sufficient number of test data is required. This is prepared for the test process in a next step (*Creation of Test Data*). The test is realized as it was defined in section 2.5.6.7: First, test data is used for the neuronal activation in a step called *Calculation of the Network's Output*. The consecutive step (*Comparison with Test Output*) establishes the generalization error \underline{E}^G as the difference between the network's output and the intended test output. If the generalization error is not appropriate for the definition of the desired test quality, the network's parameters are adjusted in a further step (*Adjustment of the Network's Parameter*). This can refer to changes in the network architecture, learning parameters, learning algorithms, etc. By making those changes, a varied training phase is started, and those cyclic processes are repeated until the test error is in accordance with the definition of the desired test quality.

Use Phase: If the developed neuronal network satisfies both the desired training and test quality, the neuronal network can be considered as appropriate for use within the specified problem and application domain. As long as those domains are not

changed, e.g. because of shifting relations or further relations of the data sets, the developed neuronal networks can be used reliably.

Intending the development of neuronal networks as a demonstration of the new artifact of the CoNM and corresponding sub-artifacts, the method presented here serves as the methodical reference point for the development of neuronal networks in the corresponding demonstration sections of this contribution.

3.2.4 Empirical Evaluation Method

Empirical research is realized with the help of a sequence of consecutive steps, each guiding the researcher to the systematic identification of decisions in regard to the intended research objective (Diekmann 2012, p. 186–230). Fig. 3.7 visualizes the required steps.

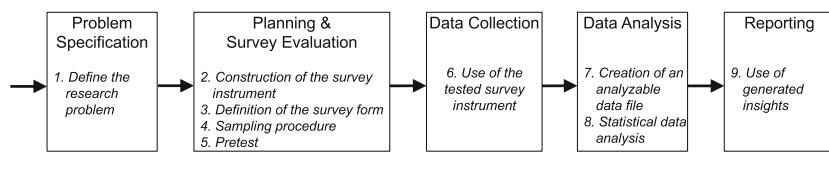


Figure 3.7 Methodological Approach for Empirical Research (Diekmann 2012, p. 186–230)

Within the figure, five process steps are visualized, which are connected by arrows indicating the *nominal process sequence*. In the following section, the five process steps are described in accordance to Diekmann (2012, p. 186–230):

Problem Specification: At the beginning, the research problem is specified as follows: On the basis of an SLR, the current state of research work is acquired, and a research gap is identified. Here, practical problems can guide this process. Then, the research objective and research questions are formulated explicitly. Finally, the survey method is chosen. Following Bortz and Döring, the following three kinds of empirical studies can be distinguished: *Explanatory studies*, which focus on the explanation of known relations and hypotheses, *exploratory studies*, which focus on the identification of new relations and hypotheses, and *descriptive studies*, which focus on the characterization of a population (Bortz and Döring 2006, p. 51, 356, 489).

Planning and Survey Evaluation: In a second step, the survey instrument is specified as follows: The survey method and form is chosen, then definitions about the operationalization are found. Following Häder, the following three kinds of survey methods are generally used (Häder 2010, p. 187): *observation*, *interview* and *content analysis*. All of them can be carried out orally, on a written base or via the Internet, which refers to the survey form. Then, the sampling procedure is addressed: Here, the composition of test person selection and the extent and period of the survey are characterized. Finally, the survey instrument can be constructed on the basis of concretized operationalizations. The pretest applies the constructed survey instrument in a small test setting, such that the survey instrument can be optimized and feed back by test persons before the main survey can be carried out. Hence, bad formulations, non-understandable instructions, and further test problems can be weeded out efficiently.

Data Collection: In a third step, the specified survey instrument is used for the purpose of data collection. For this, required interview teams are educated and a denial rate minimizing introduction letter, presentation, etc. is presented. During the survey process, a quality assurance is carried out.

Data Analysis: Data that has been collected during the survey process is stored in a structured data collection (Schnell, Hill, and Esser 2011, p. 9). In order to assure a high data quality, which is the foundation for good analyses, the following points are considered: *Wild codes* are adjusted, meaning answers that do not fall within the permitted scale are mapped to special categories such as “no information”, “out of range”, etc., with *implausible* and *inconsistent* data being excluded. Further, *missing values* are identified and treated individually (Gronau and Grum 2016, p. 34). Considering those adjustments, a data file that can be used in further data analyses is prepared. Often, in this stage, statistical data analysis methods are carried out with tools such as Excel, SPSS, Python, etc. Based on the analysis results, research insights can be generated, which have to be integrated in and harmonized with the theoretical foundation of the research object.

Reporting: The fifth and last step is connected to the communication of research insights. For this, the content-based proceedings and results of the research project are documented in a research report. Graphical illustrations shall simplify the understanding. Further, recommendations shall enable a practical use of research insights.

Intending the application of the CoNM in real-world scenarios as a demonstration, the method presented here serves as a methodical reference point for any empirical research in the corresponding demonstration sections of this contribution.

3.2.5 Morphological Box

Although the construction of a morphological box demands an individual research process, such as the general morphological analysis (Zwicky 1966), in this research, its construction has been structured by the proceeding of section 3.2.4. So, the architectural design of the software called *Augmentor* (see section 5.3.1) was realized in the CoNM context under the *Empirical Evaluation Method*, because the identification of the acceptance of test persons' demands for an empirical analysis and the data conductance was based on a survey instrument.

3.2.6 Mixed Methods Design

Because of the observation that research problems increase in complexity, isolated method uses cannot generate sufficient explanatory power (Cresswell 2014, p. 13–26). As different methods are combined systematically in order to deal with a complex problem, one speaks from *mixed methods design* (Kuckartz 2014, p. 33). This can follow one or several phases and requires the integration of results and conclusions of separate methodical approaches. Since even contrary approaches, such as qualitative and quantitative approaches (Weber 2015, p. 84), can be combined, research questions can be answered with hard facts such as numbers and key performance indicators or soft facts such as argumentative statements. Since contrary approaches, such as behavioristic methods and construction-based methods, can be combined (Becker and Pfeiffer 2006), different kind of artifacts can stand side by side.

Since the “Concept of a Neuronal Modeling” is a complex artifact as well, some sub-artifacts serve as software expansion. Others design theoretical models or refer to experiments and practical models. Further sub-artifacts design algorithmic and mathematical formulations, etc. Hence, a single method is not capable of standing in for all sub-artifacts. Therefore, a common proceeding model for the design, demonstration, and evaluation of the CoNM will be a combination of various methods and will follow a mixed methods design.

3.3 Methodical Approach

As has already been clarified in the introduction chapter, the construction of a CoNM is challenging because of its innovativeness as the main artifact and its great number of heterogeneous sub-artifacts. Each artifact has particular requirements related to the research approach.

The following requirements were related to the methodical approach:

- The methodology intends to follow a design-oriented research approach.
- Each sub-artifact shows its individual requirements regarding the research approach: There are standing NN experiments next to empirical examinations. Further, there are mathematical and algorithmic definitions besides software design and implementations. Therefore, various, specific methods are required.
- The sub-artifacts have to be considered within their specific context. A greater context is deemed necessary as well. Therefore, a mixed method design is an attractive choice.
- Both the relation between the sub-artifacts and that between the main artifact and its sub-artifacts are intended to influence each other. Therefore, a mixed method design is required.

The methodological approach used in this contribution can be seen in Fig. 3.8. It is described in the following section.

Since the construction of the main artifact—the CoNM—is intended to be realized in a design-oriented manner, the main proceeding follows the DSRM (section 3.2.2). The complexity is reduced by identifying sub-research questions and atomic artifacts, which are mapped to three methodological levels combined in a mixed method design (section 3.2.6). The starting point for each artifact is an individual problem definition.

The first methodological level deals with the foundation and research gap identification of the main research project including all its sub-artifacts. One main SLR aims to identify relevant, high-quality contributions that ideally have provided quantitative evidence, the so-called body of knowledge of this contribution. Since the body of knowledge affects the design of all sub-artifacts and insights made because of the development of the sub-artifacts' demand for an SLR update, the SLR is integrated by an iterative control flow within the main DSRM. So, the main artifact and all sub-artifacts can be placed precisely within the current state of knowledge, and a communication can be carried out properly.

The second methodological level deals with sub-artifacts that can be realized with the help of its own DSRM. An evaluation ensures either a modification of a solution's objectives or a modification in regard to the design and development of that artifact until its individual problem is resolved. Then, the sub-artifact is considered within the CoNM context.

The third methodological level deals with sub-artifacts that contain NN experiments. It is oriented towards Lämmel and Cleve (cf. section 3.2.3), who show the training and testing of NN. Since it is nested within the DSRM, it is ensured that

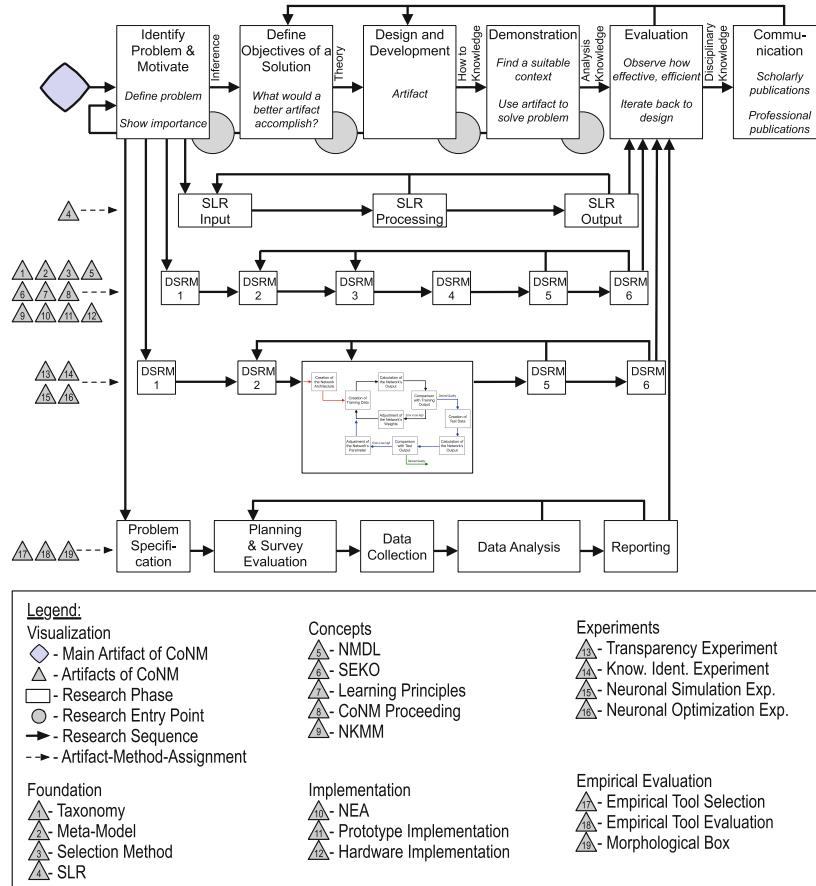


Figure 3.8 Methodological Approach for the Dissertation

those experiments do serve their greater demonstration context. When experiments are finalized, they are considered within their greater CoNM context.

The fourth methodological level deals with empirical sub-artifacts, which can be realized following Diekmann (cf. section 3.2.4). In accordance with the first two methodological levels, a cyclic proceeding is realized with the help of recurrent process connections. Those recurrent connections visualize the possibility to restart

the empirical research because of later insights. Either the data analysis or the reporting of the sub-artifact can lead to an adjustment, such that experimentations fit into the main artifact context.

Since Peffers et al. stated that the phases of the DSRM do not proceed in sequential order from the first to the sixth phase (Peffers et al. 2007, p. 14), inferences from the evaluations of the sub-artifacts and the main artifact have to be considered. For example, the empirical feedback of the software evaluation can be the cause for necessary software design modifications and its corresponding implementation. Therefore, the results presented here can be interpreted as the result of many attempts and repeated runs of the phases within the methodological approach.



Design

4

The design considers the problem analysis and the identified research gap of chapter 2 and examines how the requirements of this contribution (see section 3.1.3) can be addressed conceptually. In accordance with Peffers et al. (2007), this refers to the third step of a design-oriented research proceeding. Faced with the design objective of this research, which refers to the construction of a *Concept of Neuronal Modeling*, the following establishes the CoNM context as follows:

In the CoNM context, ANN are considered as models (Def. 1) that have a processual meaning (Def. 3). This evolves because of the presence of knowledge (Def. 13), which becomes real in the iterative interplay of concrescence in the private and satisfaction in the public. Hence, the smallest ANN-based system is recognized as the most atomic knowledge carrier.

Since the ANN's knowledge provides a temporal coherence and unity and manifests because of the intended application as consequent value creation from the viewpoint of an observer, any form of objects can be approximated by ANN and can be interpreted as processual components or rather as a composition of actual entities. These participate in the iterative interplay of concrescence in the private and satisfaction in the public by the consequent knowledge application, which can be measured by right tools. In other words, the world constructs because of the knowledge application by Mother Nature – or rather “Mother Actual Entity”, which is referred here to as knowledge construction axiom (Def. 1). Hence, its rebuilding by ANN-based systems is assumed to be based on the composition of neuronal structures and each neuron represents an actual entity.

Supplementary Information The online version contains supplementary material, such as Appendix A, B, C and D whose references are indicated in the text, available at https://doi.org/10.1007/978-3-658-35999-7_4.

Being faced with the processual definitions of process models (Def. 3), business process models (Def. 4), knowledge transfer models (Def. 14) and simulation models (Def. 16), the *neuronal process model* brings neuronal models (Def. 24) and processual understandings together and it is defined to be the following:

Definition 32 (Neuronal Process Model).

A neuronal process model is a process model of at least one process or a hierarchy of processes, while each process refers to overlapping, independent actual entities providing a temporal coherence and unity about the identification of knowledge, about its transfer from knowledge carrier to knowledge receiver, and about its application by the knowledge receiver, can be measured by right tools, are logically from the perspective of an individual as it is understood, form all together the actual existing world in a multiplicity and become real in the iterative interplay of concrescence in the private and satisfaction in the public, and is realized with the aid of biologic plausible mechanisms inspired from the human brain.

Accepting more or less complex ANN as set of system elements se representing processual systems being connected by a set of relations sr , the use of its mathematical representation x refers to the realization of simulation models (compare with simulation model Def. 16). Hence, the neuronal simulation imitates the iterative interplay of concrescence in the private and satisfaction in the public by the use of x (compare with neuron model Def. 24).

The manual model construction as well as the application of learning principles on behalf of a reproduction of the functioning of the object system y , which is the human brain, therefore is considered as the neuronal modeling of any kind of processes or rather process models because it transfers the models from inaccurate in simulations to a state of being more accurate (close to reality) in the same simulations.

The passage to speak about optimization of models is rather vague, since manual modifications as well as the application of learning principles are comparable to the act of modeling neuronal processes. They differentiate in the point that any kind of valid model is considered as starting point: These are addressed by modeling activities, but following the intention to make the process more effective, which worsens the neuronal simulation accuracy in the sense that it represents a to-be state (far from reality). One also can say that the optimization increases the simulation accuracy of the desirable state of a to-be reality.

In this context, sub-artifacts of the CoNM are presented section-wise: The first sub-section presents a taxonomy and derives a meta-model, that provides objects being relevant for the CoNM. The second sub-section presents a method for the iden-

tification of an appropriate foundation of the CoNM. This addresses software, hardware, concepts, methods and any relevant form of tools. Further, this method will be used for the evaluation of the CoNM. The third sub-section presents an architecture, which addresses the linkage of the CoNM in companies. The fourth sub-section presents a proceeding that guides the CoNM use. The fifth sub-section presents a management approach, with whom different kinds of experts are coordinated at the CoNM use. The sixth sub-section conceptualizes the technical foundation for the CoNM and presents the modeling language, which serves as communication platform of human as well as machine-based model constructors. The seventh and eighth sub-section consider this modeling language and present approaches for the exhaustion of knowledge from ANNs as well as learning principles in the context of CoNM. Jointly, these sub-artifacts will manifest the *Concept of Neuronal Modeling* and they compose to the main artifact called CoNM.

4.1 Taxonomy and Meta-Model Creation

The selection of appropriate techniques, such as a modeling language, a process modeling and simulation tool and an ANN library, is realized on behalf of an ANN process taxonomy and a generic meta-model for the ANN process domain, which are designed in this section. The design process requires the taxonomy to be designed before the meta-model and is explained in the following.

The generic meta-model is derived from theory and terminology of both the process domain and ANN domain. It can be used in two ways: On one hand, meta-models of available modeling languages, tools and libraries can be compared with the meta-model for the ANN process domain presented here. This would focus on meta-model elements and requires similarity measures in regard to meta-model elements. Hence, those approaches would be most attractive whose meta-model distinguishes only little from the here-presented meta-model. Since only few modeling languages provide meta-models, this is not very promising (Scheer, 1998; List and Korherr, 2006, p. 1535; Grum and Gronau, 2018b). On the other hand, a generic meta-model can be used for the identification of required graphical notation elements (modeling languages) or software functions (tools and libraries) to explicitly symbolize a certain element of the generic meta-model. This requires similarity measures in regard to graphical notation elements and software functions. Hence, those approaches would be the most appealing that provide most graphical notation elements and software functions representing the meta-model presented here. Since attractive modeling languages, tools and programming libraries shall not be excluded from the selection process (see requirements of section 3.1.3) only because of a missing meta-model, the comparison will focus on the second option.

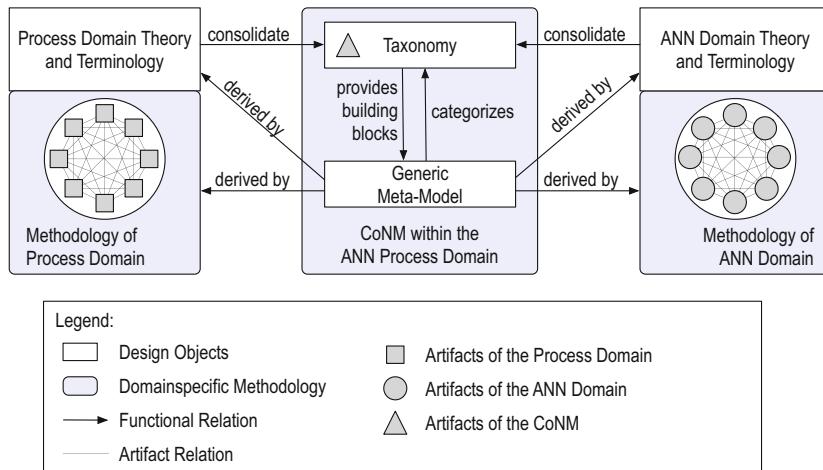


Figure 4.1 The Design of the Meta-Model Creation

In accordance to Giaglis, a taxonomy is a good foundation for the selection of appropriate techniques to use (Giaglis, 2001, p. 15) and comparisons on meta-concepts are widely accepted (Scheer, 2002, p. 132). The here-established ANN process domain consolidates theory and terminology of both the process domain and ANN domain. Hence, it specifies the system border (Krallmann, Bobrik, and Levina, 2013), provides building blocks to the generic meta-model and helps to categorize meta-model elements. The design process is visualized in Fig. 4.1.

Faced with design dependencies of the artifacts of the ANN process taxonomy and the generic meta-model for the ANN process domain, the designed taxonomy will be presented in a first sub-section. A second sub-section specifies the design of the generic meta-model. Therefore, based on this specification, the generic meta-model is designed in a third sub-section.

4.1.1 ANN Process Taxonomy

The ANN process taxonomy presented here is designed to join perspective-oriented taxonomies. As not one taxonomy from literature considers the neuronal level (see research gap of section 2.1.2), the ANN process taxonomy presents an extension. Although the there-presented collection of perspectives and taxonomies of literature was created with the intention to present an extensive picture, this collection cannot

raise a claim for completeness. Nevertheless, it is a good foundation for the selection of appropriate techniques to use (Giaglis, 2001, p.15).

In accordance with Frank (1993, p.167), a selection of perspectives is never objective and free from subjective attitudes, even after careful consideration and reasoning. In order to deal with this subjectivity and create a taxonomy close to reality, the here-presented taxonomy follows the recommendation of Wollnik and Gochla (1986): It only builds on perspectives, which have been evaluated to be relevant by a greater number of persons (Wollnik and Gochla, 1986, p.44).

Faced with requirements of section 3.1.3, beside the consideration of socially accepted perspectives, the consideration of business process, knowledge, and neuronal perspectives is essential. Hence, the following perspectives are used for the integration of relevant process elements:

- business process context perspective
- behavioral perspective
- functional perspective
- informational perspective
- knowledge perspective
- organizational perspective
- neuronal perspective
- process ANN perspective
- simulation perspective

All of those perspectives are required for the design of the NPM, NPS, and NPO (see Fig. 3.1) because of the following:

1. A process modeling cannot be carried out without knowing organizational entities.
2. A simulation of processes cannot be realized without the behavioral specification and the characterization of material, information, and knowledge flows.
3. An optimization cannot be evaluated without the corresponding business context.
4. Real, biologic models of human- and non-human-based brains cannot be used within the process domain without being integrated.

Hence, the interdependencies of those perspectives become clear and selected perspectives are to be considered within one, integrated model. The structuring of each perspective will therefore be developed individually in the following sub-sections. Then, the relation of the perspectives will be considered, so that single perspectives are integrated.

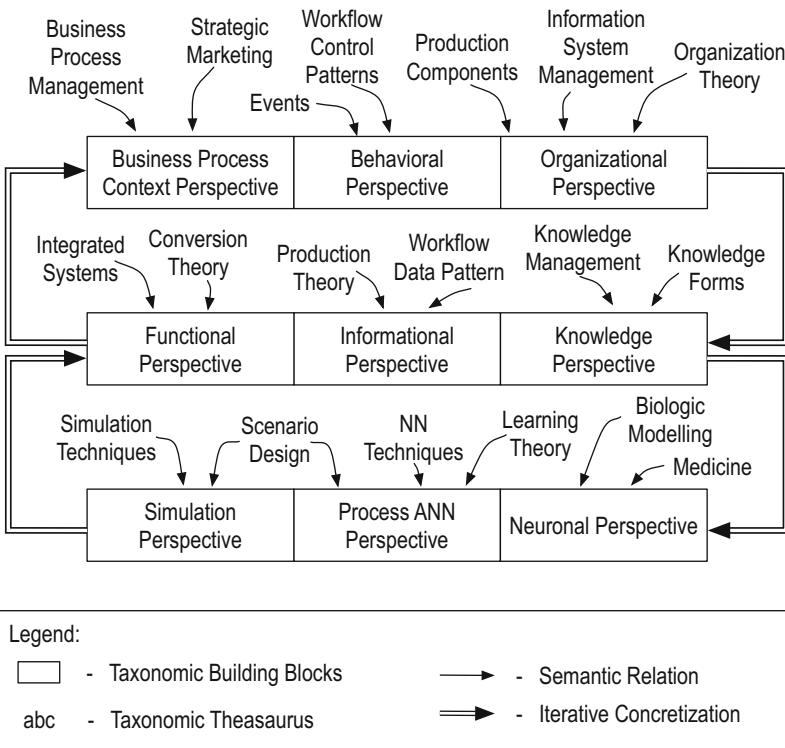


Figure 4.2 Iterative Concretion of Perspectives by the Introduction of Central Terms and Definitions

Faced with the complexity of each perspective and model, the desire to achieve a final level of structures and relations of perspectives within this contribution is not realistic. Hence, the here-presented level of detail intends to create a foundation for discourses about *multi-perspective neuronal enterprise models*. The exemplary concretion of each perspective by the introduction of central terms and definitions as it will be used in the following sub-sections can be seen in Fig. 4.2.

The problem that Osterweil (1987) pointed out as *chicken-and-egg problem* in exploring representations for process modeling languages is visualized by double-lined arrows in Fig. 4.2. It means the following: In order to find out what language features are needed, the language has to be applied. Furthermore, in order to apply a modeling language, appropriate language features are needed. Hence, an iterative approach with this contribution will be the best start.

Although the theoretical foundation provided further perspectives, such as the activity perspective, communication perspective, strategic perspective, role perspective, service-oriented perspective, and speech-act oriented perspective, the here presented taxonomy follows the philosophy that perspectives are intersecting and some provide part-of-relationships. For this purpose, the definition of views is specified in contrast to the perspective definition (see Def. 6), which can reduce the complexity of perspectives to an arbitrary level. This is in accordance with the literature, since each view reduces the complexity of many entities of a perspective to a selection of only some entities (Scheer, 2002, p. 137).

Definition 33 (View).

A view captures any perspective on an original or model system from the viewpoint of one or more stakeholders or ANN instances, in a certain notation, and for a certain purpose.

In harmony with the literature reviews, the act of modeling is drastically simplified by this, since the construction of a view on a system is allowed to be incomplete and to result because of multiple models (Rolland, Prakash, and Benjamen, 1999). Hence, a view concept can therefore help to combine alternative, simple modeling methods in order to collect or simply visualize required information. Integrated by a common meta-concept, individual views then can be integrated to one error-poor, consistent, redundancy-free model representing the reality (Zhang, Beetz, and Vries, 2013). Following this perspective-view distinction, it is possible to create one integrated, coherent *multi-perspective neuronal enterprise model*, but only visualizing required views. Hence, the following presents a mixture of perspectives. The titles or rather their denominations have been derived from only that views, that contribute most.

4.1.1.1 Business Process Context Perspective

The business process context perspective is based on terminologies, theory, and methods of the business process management, strategic marketing (see Fig. 4.2) and is defined to be the following:

Definition 34 (Business Process Context Perspective).

A business process context perspective captures the conceptual model of the process's business context from the viewpoint of one or more stakeholders, in a certain notation, for the purpose of maximizing the effectiveness and efficiency of the realization of business processes in regard to the customer.

Building Block Description: In accordance with this definition, the here-presented perspective provides a notation for modeling a goal-oriented economization of companies with help of business processes.

Business processes focus on the value for *customers* and each is connected to a *process owner* (Hammer, 1996). This person is responsible for satisfying the customer with by the process delivered *products and services* (Jacobson, Ericsson, and Jacobson, 1995). Faced with the type of value generation, business processes can be distinguished to be one of the following business process types: *core processes*, *support processes* or *management processes* (Ould, 1995). While the first is independent from or supported by support processes, the second focuses on indirect value generation issues and supports core processes. The third deals with the controlling of the first two business process types.

Each *business process* can be evaluated by *qualitative measures* and *quantitative measures* (Harrington, 1991). This is realized on base of a comparison of measured *to be values* (planned, nominal values) and *as is values* (unplanned, real-life values). Through this, it becomes clear if the corresponding business process achieves its *process goals* and so supports one of the main *enterprise goals* (Thompson and McEwen, 1958; Kueng and Kawalek, 1997; Rolland, Souveyet, and Achour, 1998; Park et al., 2017).

Relevance: The integration of this perspective is essential for the CoNM because of the following:

A fast understanding shall be guaranteed for people who do not know or need to know relevant business theory in detail. An understanding is achieved although because relevant relations are made transparent in this process model.

Further, the fast understanding is supported for people who do not know or need to know relevant theory of further perspectives in detail. An understanding is achieved without working through the complex process logic because irrelevant relations are separated from this process model but integrated although in the background.

Since it can be clarified how business processes support the achievement of goals without knowing the entire collection of complex process models, corresponding perspectives and underlying technical concepts, which show how and under which conditions this is realized, the effectiveness and efficiency of business processes is supported.

Further, it can be specified how the business context is reflected in the simulation perspective and process ANN perspective without knowing the entire collection of complex process models, other perspectives and underlying technical concepts. Hence, a target-oriented optimization is supported.

4.1.1.2 Behavioral Perspective

The behavioral perspective is based on terminologies, theory, and methods of workflow control patterns, events, tasks (see Fig. 4.2) and is defined to be the following:

Definition 35 (Behavioral Perspective).

A behavioral perspective captures the conceptual model of the performance of process elements from the viewpoint of one or more process owners, in a certain notation, for the purpose of guaranteeing the functioning of timing, decision, quality, and proceeding issues of processes.

Building Block Description: In accordance with this definition, the here-presented perspective provides a notation for modeling the flow of data, knowledge and material from task to task, which aggregate to a certain business process. Hence, it describes how and when tasks of a business process are performed, e.g. through sequences, iterations, feedback loops, entry and exit criteria, complex decision making conditions (Curtis, Kellner, and Over, 1992).

A task can either be an *atomic task* or be composed by further *sub-tasks*, which are recursively refined by tasks. The realization of a task can be triggered by *events*. Since it provides *time-based states* (Workflow-Management-Coalition, 1998, p. 9), the behavior of the business process can be controlled as follows: The *sequence* is visualized by a *control flow*, basic workflow control patterns are visualized by *control nodes (and split, and join, xor split, xor join)* and different types of synchronization are visualized by *or join, or split* and *n-out-of-m* (Aalst et al., 2003). Similar to List and List and Korherr (2006, p. 1533), the *data flow, information flow* and *knowledge flow* connect atomic activities and indicate in the first case the flow of data objects, in the second case the flow of tangible and non-tangible material objects and in the third case the flow of knowledge and information objects.

Relevance: The integration of this perspective is essential for the CoNM because of the following:

A fast understanding shall be guaranteed for people who do not know or need to know theory relevant for the behavior of systems in detail. An understanding is achieved although because relevant relations are made transparent in this process model.

Further, the fast understanding is supported for people who do not know or need to know relevant knowledge management, creation, analytics, product engineering theory of further perspectives in detail. An understanding is achieved without working through the complex process logic because irrelevant relations are separated from this process model although they are integrated in the background.

So, it can be clarified in which order and under which conditions tasks are performed without knowing the entire collection of complex process models, other perspectives and underlying technical concepts. Hence, the functioning of processes is supported.

4.1.1.3 Functional Perspective

The functional perspective is based on terminologies, theory, and methods of integrated systems, knowledge transfer theory (see Fig. 4.2) and is defined to be the following:

Definition 36 (Functional Perspective).

A functional perspective captures the conceptual model of the performance of process elements from the viewpoint of one or more process participants, in a certain notation, for the purpose of guaranteeing the availability and provision of for the performance of consecutive process element required entities.

Building Block Description: In accordance with this definition, the here-presented perspective provides a notation for modeling the data, material, and knowledge creation (and elimination) on base of available entities, such as information resources, knowledge resources and traditional resources. This includes learning, forgetting, synthesis, analysis, decomposition, etc. With this, the provision of process entities for consecutive tasks is guaranteed and the performance of process elements is carried out (Curtis, Kellner, and Over, 1992).

An *activity* can either be an *atomic activity* or be composed by further *sub-activities*. Those are refined recursively by tasks (List and Korherr, 2006, p. 1533). Although a sequential and parallel visualization of activities can be created, a control flow and timing issues are not specified here rather than in the behavioral perspective. Hence, activities are performed at least one time but can be accessed repetitively. Hence, the identification of *events* can either come from process owners or inductively be triggered by the activity itself. The working of activities is connected to the fulfillment of *conversion requirements* and is carried out by *conversion methods* (Gronau and Maasdorp, 2016, p. 17).

Since real world objects provide a position within the real world that changes over time, the realization of an atomic activity provides a unique time-based position, which is derived by the active process participant. When activities are grouped, the *time-based position* can be derived by corresponding process participants and resources and focuses on the working and optimal bringing together of them and required resources.

For the realization of activities, *resources* are consumed and produced (Kuong et al., 1996, p. 17). Hence, activities are connected with resources as *input objects* and *output objects* (Krallmann, Bobrik, and Levina, 2013). The value creation is operationalized on base of the SECI-model (Nonaka and Takeuchi, 1995). Here, four kinds of *conversions* (*internalization*, *externalization*, *combination*, *socialization*) conceptualize the value creation.

Although the SECI-model was originally meant as concept for knowledge creation, intentional material and data manipulations incl. their transfers are interpreted as combinations, where the corresponding objects are enriched to explicit knowledge in interpreting them within the context of the current activity. This refers to North's requirement of the enrichment by a *meaning* and *contextual embedding* (North, 1998, p. 41) and follows Grum's and Gronau's *knowledge construction axiom* (Grum and Gronau, 2018a). Following this axiom, various kinds of commonly accepted flows (see section 2.1.1.3) can be reinterpreted as conversion, such as the following: functional flows, performance flows, information flows, organizational flows, objective flows, control flows, resource flows, financial flows, etc., which all can follow contrary directions (Scheer, 2002, p. 11–31). Since ANN are not able to distinguish between all those kinds of flows on an atomic neuronal level and simulated neuronal networks only consider one kind of activation, this simplification is essential for the construction of the CoNM.

Axiom 1 ::= KNOWLEDGE CONSTRUCTION AXIOM

Commonly accepted kinds of process flows describe a value creation, which is based on the intended application of knowledge concepts and can therefore be operationalized by the SECI-model.

Enrichments based on this axiom can be carried out either by CPS or information systems that have been enabled through information objects, such as applications, transportation orders, etc. Alternatively, they have been enabled by humans or CPS through knowledge objects, such as the intended production, transportation by themselves, etc. Non-intentional conversions, such as real-world processes (e.g. weather) and physical laws (e.g. gravity) are neglected based on the following interpretation: Even the interpretation of real world processes are created with intention and only by the subjective perception of the individual. Here, the observation of physical phenomena (as is values are collected) and its comparisons with to be values, which are based on the belief and expectation of the individual, leads to the adjustment of the individual's beliefs and their externalization to physical laws. Hence, their use within process models is always connected to the use of information objects. The identification of different kinds of process flows are therefore interpretations,

which are realized on higher cognitive levels than atomic neuronal levels. Hence, their identification can be carried out in a second step.

Relevance: The integration of this perspective is essential for the CoNM because of the following:

A fast understanding shall be guaranteed for people who do not know or need to know relevant ANN theory in detail. An understanding is achieved in working through hierarchically decomposed process logic because relevant relations are made transparent in this process model.

Further, it can be clarified how objects required for tasks are created without knowing the entire collection of complex process models, corresponding perspectives and underlying technical concepts. Hence, the availability and provision of entities is supported.

4.1.1.4 Informational Perspective

The informational perspective is based on terminologies, theory and methods of the production theory, workflow data pattern (see Fig. 4.2) and is defined to be the following:

Definition 37 (Informational Perspective).

An informational perspective captures the conceptual model of traditional and informational entities manipulated and created during the performance of activities from the viewpoint of one or more organization managers or information managers, in a certain notation, for the purpose of providing them as organizational entities.

Building Block Description: In accordance with this definition, the here-presented perspective provides a notation for modeling elements being produced or consumed by an activity, such as data, intermediate artifacts, end products and objects (Curtis, Kellner, and Over, 1992).

In general, this refers to *traditional resources* and *informational resources*. Both have been integrated in one perspective since traditional resources are combined with a digital twin and are essential for simulations (Boschert and Rosen, 2016). Hence, a physical *time-based position* is connected to material objects and information objects.

Inspired by the input-output-view of integrated systems (Scheer, 1999), *traditional resources* can be either *tangible*, such as products, materials, physical objects, etc., or they can be *non-tangible*, such as services, consultations, held presentations, etc. All are visualized by *material objects* and are stored in a *materialbase table*. The latter can stand for example for a storage room, facility, containers, etc.

Inspired by workflow data patterns (Russell et al., 2004) and similar to List and Korherr (2006, p. 1533), environmental information resources refer to three types: *data repositories*, *applications* and *services*. While the first stores *data objects*, such as the digitalization of physical objects, meta-data about physical objects, a stream of production machine data, documents. etc., the latter two can be categorized to *information systems*.

Since both, traditional and informational resources have a lifespan (Fröming, 2009; Tao et al., 2018), their life-cycle is essential and is visualized as well.

Relevance: The integration of this perspective is essential for the CoNM because of the following:

A fast understanding shall be guaranteed for people who do not know or need to know relevant ANN theory and functional perspective relevant theory in detail. An understanding is achieved in working through hierarchically decomposed material and data organization logic because relevant relations are made transparent in this process model.

Further, it can be clarified which material and data objects for tasks are available or shall be available without knowing the entire collection of complex process models, corresponding perspectives and underlying technical concepts. Hence, the availability and provision of entities is guaranteed.

4.1.1.5 Knowledge Perspective

The knowledge perspective is based on terminologies, theory and methods of strategic and operative knowledge management, knowledge forms (see Fig. 4.2) and is defined to be the following:

Definition 38 (Knowledge Perspective).

An knowledge perspective captures the conceptual model of knowledge entities created during the performance of activities from the viewpoint of one or more knowledge managers, in a certain notation, for the purpose of providing them as organizational entities.

Building Block Description: In accordance with this definition, the here-presented perspective provides a notation for elements being produced or consumed by an activity. As knowledge resource is termed the unity of skills, cognition and capabilities, which are used by individuals for the solution of given problems (Davenport and Prusak, 2000; Krogh, Ichijo, and Nonaka, 2000; Polanyi and Sen, 2009).

Inspired by the previously mentioned SECI-model, knowledge is differentiated to tacit knowledge and explicit knowledge (Nonaka and Takeuchi, 1995). Both have

been integrated in one perspective since both kinds differ fundamentally: The first kind of knowledge can be externalized easily and stored in *informationbase tables* since it is not connected to persons or CPS as *knowledge carriers* anymore. Here, the person independent *information repository* serves as knowledge carrier. This kind of knowledge is visualized by *information objects*. The second kind of knowledge can not be externalized and is connected to humans or CPS as *knowledge carriers*. This kind of knowledge is visualized by *knowledge objects*. Since it can not be externalized, a *knowledge description* intends to verbalize the corresponding knowledge. A *knowledgebase table* stores meta-information about knowledge objects and serves for its organization.

Both kinds of knowledge are only valid in their corresponding *knowledge context* and can be categorized by *competency classes* (Gronau and Maasdorp, 2016, p. 17). Since both are connected to physical objects, a *physical position* is connected to information objects and knowledge objects.

Relevance: The integration of this perspective is essential for the CoNM because of the following:

A fast understanding shall be guaranteed for people who do not know or need to know relevant ANN theory and functional perspective relevant theory in detail. An understanding is achieved in working through hierarchically decomposed knowledge organization logic because relevant relations are made transparent in this process model.

Further, it can be clarified which knowledge and information objects for tasks are available or shall be available without knowing the entire collection of complex process models, corresponding perspectives and underlying technical concepts. Hence, the availability and provision of entities is guaranteed.

4.1.1.6 Organizational Perspective

The organizational perspective is based on terminologies, theory, and methods of information system management, production components, organization theory (see Fig. 4.2) and is defined to be the following:

Definition 39 (Organizational Perspective).

An organizational perspective captures the conceptual model of organizational units required for the performance of process elements from the viewpoint of one or more organization managers, in a certain notation, for the purpose of guaranteeing the best compromise of available and required entities.

Building Block Description: In accordance with this definition, the here-presented perspective provides a notation for elements required for the production or consumption of resources during the performance of an activity.

In general, an activity is carried out by at least one *process participant*. This demands for and supplies resources, which includes all kinds of resource objects and considers process participants as knowledge carriers (Gronau and Maasdorp, 2016, p. 17).

Inspired by the Workflow Management Coalition, the following types of workflow participants are considered as process participants (Workflow-Management-Coalition, 1998, p. 17): *humans* and automatic resources, which refer in state-of-the-art systems to *information systems (software applications, software services)* and *CPS* (Grum and Gronau, 2018b). Hence, those can have an organizational *role* and are connected to an *organizational unit* (List and Korherr, 2006), which either is *internal* and part of the own company or *external* (Ullrich et al., 2017).

Since real world objects provide a *time-based position* within the real world, the realization of an activity is dependent on the working bringing together of its process participants (Grum and Gronau, 2018b).

For the successful realization of activities, activity specific requirements have to be met by process participant specific *requirements*, which refers e.g. to its abilities, willingness, organizational permissions, motivation, competence (North, Brandner, and Thomas Steininger, 2016, p. 48).

Relevance: The integration of this perspective is essential for the CoNM because of the following:

Since more and more activities are performed by autonomous systems, the considering of state-of-the-art production components, such as robots, CPS, intelligent computer systems, is essential. The system's capabilities to manipulate real world objects and generate knowledge similar to humans will improve and increase their contribution to the value-generation in business processes. As the bringing together of real world objects shall be optimized similar to the functioning of the human brain, this perspective provides a fundamental part or the reorganization of the position of objects within the real world and their functional connection. Hence, their consideration will play an essential role in the NPM, NPS, and NPO.

4.1.1.7 Neuronal Perspective

The neuronal perspective is based on terminologies, theory, and methods of biologic modeling, medicine (see Fig. 4.2) and is defined to be the following:

Definition 40 (Neuronal Perspective).

A neuronal perspective captures the conceptual model of the working of human- and non-human-based brains from the viewpoint of one or more neuroscientist and medical doctor, in a certain notation, and for the purpose of providing strengths of biological models for contexts of further perspectives.

Building Block Description: In accordance with this definition, the here-presented perspective provides a notation for modeling the very basic task realization on a neuronal level and provides the effective and efficient organization capability of biologic models being optimized over centuries by evolution.

In general, brains consist of the connection of billions of neurons (Azevedo et al., 2009), whose neuron type can be derived by their location within the body and their corresponding lengths (Schalkoff, 1997). Neurons refer to the following: the neuron's cell body, the cell's nucleus and cell plasma, which is bordered by the cell body and holds the nucleus (Snell, 2010, p. 34). While the cell nucleus shows importance for the heredity, e.g. which kind of neuron will be created, the plasma holds the molecular equipment, which is required for the working (e.g. the signal transmission) of the neuron (Jain, Mao, and Mohiuddin, 1996). Since the kind of neurons won't be differentiated in this contribution, yet, all metabolic processes of the neuron are taken together and modeled by a *neuron*. Those lead to a signal transmission behavior, which is modeled by an *activation function*, such as *entity*, *sigmoid*, *logistic*, *threshold* etc. (McCulloch and Pitts, 1943). The neuronal connection is realized by the following kinds of *cell protuberances*. Afferent cell protuberances, which refer to areas receiving signals, represent *dendrites*. Efferent cell protuberances refer to areas forwarding signals. Those include *nerve fibers* and *synapses* (Basheer and Hajmeer, 2000). While the first refer to axons (= neurites), that are surrounded by glial cells (= myelin sheaths) and branch into collaterals, the latter refer to either chemical or electric synapses (Alcamo and Bergdahl, 2003, p. 100). For a first simplification, all of them are modeled by one connection *weight*. The integration of incoming signals is carried out by the *axon hill* of a neuron. This is modeled by an *aggregation function*, such as the *summation*, *multiplication*, etc. (Haykin, 1994). The axon hill is the place of the creation of an action potential, which refers to a temporal and neuron specific difference of a cell's membrane potential and its rest potential (Snell, 2010, p. 84). In accordance to Haykin (1994), this can either be inhibitory or excitatory. Its current *fire rate* (= *currents*) is modeled by a *neuronal activation*. Hence, a neuron's action potential refers to a *neuronal output* and a consecutive *neuronal input*. The corresponding *transfer function* then jointly abstracts the behavior of a neuronal system and so includes the modeling by weights, the aggregation function, and the activation function, so that an arbitrary

complex input signal of a certain system can be transferred to an arbitrary complex output signal by the activity of that system.

Relevance: The integration of this perspective is essential for the CoNM because of the following:

A fast understanding shall be guaranteed for people who do not know or need to know relevant neurobiology and medicine theory in detail. An understanding is achieved in working through neuronal models because relevant relations are made transparent in this process model.

Further, it can be clarified which neuronal elements for tasks are available and are active without knowing the entire collection of complex business process models, corresponding perspectives and underlying business contexts. Hence, the availability of biologic entities and their modeled behavior is guaranteed.

4.1.1.8 Process ANN Perspective

The process ANN perspective is based on terminologies, theory, and methods of NN techniques, learning theory, scenario design (see Fig. 4.2) and is defined to be the following:

Definition 41 (Process ANN Perspective).

A process ANN perspective captures the conceptual model of neuronal processing from the viewpoint of one or more data scientists, in a certain notation, and for the purpose of enabling further perspective contexts.

Building Block Description: In accordance with this definition, the here-presented perspective provides a notation for modeling the neuronal working behavior in business process contexts. It connects the process domain and ANN domain. Through this integration, the effective and efficient organization capabilities of biologic models being optimized over centuries by evolution shall be provided to the domain of process modeling. Further, business organization strategies, pedagogical learning theories, etc. shall enable the domain of AI and ANN.

The *data set* implicitly specifies the *learning context* of neuronal networks. In providing single *moments* or entire *sequences*, the data set must be prepared in a special format for *unsupervised*, *supervised* and *reinforced* learning approaches. The former two both require an *input*, while the second additionally asks for an *output*. The latter must consist of a *state*, *action* and *reward*. The learning of a neuronal network is carried out by a *trainer* in using the data set and strategy regarding a data set preparation, weight initialization, training, test and learning strategy as follows: The *data preparation strategy* specifies, how data will be prepared sup-

port an efficient learning procedure, which includes a normalization e.g. (Joost and Schiffmann, 1998). A *weight initialization* strategy refers to the question, how initial weights of a neuronal network are specified (Graves, 2012). The *stopping strategy* determines when a training can be stopped since it satisfies a stopping criteria, such as a maximal number of learning iterations, converging performance, etc. The *training strategy* specifies, which kind of learning is carried out, such as *batch learning* or *online learning* (LeCun et al., 1998). The *test strategy* specifies how the data set(s) are divided into training data set(s) and test data set(s). The *learning strategy* specifies by which technique weight adjustments within the neuronal network will be determined and the learning process will be carried out, such as *backpropagation* and *backpropagation through time*, etc. (Plaut, Nowlan, and Hinton, 1986; (Bishop, 1995); Eck and Schmidhuber, 2002a; Eck and Schmidhuber, 2002b; Graves, 2012). Because of the learning procedure, the *neuronal performance* of a neuron that carries out an *atomic activity* or a neuronal network that carries out *sub-neuronal activities*, is improved such that for example an approximation error is reduced. This can be connected to the *training error*, *test error*, *generalization error*, etc. (Bishop, 1995). Faced with the refinement of *neuronal activities*, different kinds of *network architectures* can be identified, such as *feedforward networks*, *recurrent networks*, *LSTM networks*, etc. (Rosenblatt, 1963; Rumelhart, Hinton, and Williams, 1986; Kohonen, 1989; Hochreiter and Schmidhuber, 1997a).

Relevance: The integration of this perspective is essential for the CoNM because of the following:

A fast understanding shall be guaranteed for people who do not know or need to know relevant Machine Learning and ANN theory in detail. An understanding is achieved in working through models because relevant relations are made transparent in this process model.

Further, it can be clarified how neuronal elements for tasks are trained and which kind of neuronal activity is performed by neuronal structures without knowing the entire collection of complex business process models, corresponding perspectives and underlying business contexts. Hence, the working of biologic entities within the process context is guaranteed.

4.1.1.9 Simulation Perspective

The simulation perspective is based on terminologies, theory and methods of simulation techniques, scenario design (see Fig. 4.2) and is defined to be the following:

Definition 42 (Simulation Perspective).

A simulation perspective captures the conceptual model of testing business processes and neuronal activities with scenarios from the viewpoint of one or more simulation owners, in a certain notation, and for the purpose of realizing simulation goals.

Building Block Description: In accordance to this definition, the here-presented perspective provides a notation for modeling the neuronal working behavior in business process simulations and testing its approximations.

The simulation realizes a *scenario* and clarifies if business processes achieve process goals under uncertainties (Mahmoud et al., 2009). For this, each scenario obtains its proper *simulation goal* and consists of either *atomic scenarios* or *sub-scenarios*. Only atomic scenarios generate for each simulation run and *simulation time step*, which can be fed into or be fed from computational models (Schwartz and Ogilvy, 1998). Adequate to the business process context perspective, a *scenario owner* is responsible for a scenario (Hammer, 1996). This includes the unique characterization of a *scenario description* inclusive its *initialization* parameters and *event-based changes*, which best follow a formal framework and can use the modeling language presented in this contribution (Mahmoud et al., 2009).

Relevance: The integration of this perspective is essential for the CoNM because of the following:

A fast understanding shall be guaranteed for people who do not know or need to know relevant simulation theory in detail. An understanding is achieved in working through simulation models and scenarios because relevant relations are made transparent in this process model.

Further, it can be clarified how data is provided for the neuronal data set creation, how neuronal elements and corresponding neuronal activities are tested and support a support process and enterprise goals without knowing the entire collection of complex business process models, corresponding perspectives and underlying business contexts. Hence, the working of biologic entities within the process context is supported.

4.1.2 Specification of Object Models

The conceptualization of the generic meta-model follows an object-oriented design, such that an object model is constructed. In accordance to Rumbaugh et al. (1991, p. 6), an object model is a static structure of objects in a system and their relationships among them, which is expressed by object diagrams:

Definition 43 (Object Diagram).

An object diagram is a graphical notation whose nodes are object classes, which are connected by arcs representing relationships among classes.

Consistent with theory, methodology and terminology, the following is derived and denominated as objects: abstractions of available artifacts, real-world items and circumstances. In compliance with Booch (1994, p. 39), the object-oriented design is defined to be a method of design that encompasses both the process of decomposition and the notation for depicting logical, physical, static, and dynamic models of a system.

Among many conceptualization approaches, such as the top-down structured designs (Sommerville, 1985, p. 68; Yourdon and Constantine, 1979; Myers, 1978; Page-Jones, 1988; Wirth, 1983; Wirth, 1986; Dahl, Dijkstra, and Hoare, 1972; and Mills, Linger, and Hevner, 1986) and data-driven designs (Jackson, 1975; Jackson, 1983; and Orr, 1971), the design of an object-oriented meta-model was chosen because of the following reasons:

1. Modeled objects are rather suited for analyses and can easily be mapped to the real world since an object-oriented modeling focuses on the natural abstraction of real objects.
2. The use of object-oriented models leads to well-structured complex systems (Booch, 1994, p. 77).
3. Abstractions of real objects support the creation of solid software concepts, so that those are useful for all the conceptualization, design and implementation, such that the object-oriented modeling supports a fast prototyping best (Frank, 1993, p. 172).
4. Object-oriented model encourage the reuse (Booch, 1994, p. 77). Hence, models are smaller, easier to interpret, and lead to cost and schedule benefits in implementation and maintenance.
5. Object-oriented models produce systems that are more resilient to change. Since this contribution is a first step in the ANN process domain, required changes because of on-building concepts are to expect.

Aspects of a dynamic and functional modeling are not required at this stage of conceptualization. Hence, a focus lays on their static structure and the here-specified object model is the foundation for an object-oriented meta-model, as it is designed in section 4.1.3. Hence, this sections presents design decisions before the meta-model is presented according to relevant objects, relevant relationships and their integration.

4.1.2.1 Object Conceptualization

While objects represent items and circumstances of the real-world and exist in time and space, concrete instances of the reality are abstracted, so that classes are characterized. In accordance to Booch, a class represents only the ‘essence’ of an object (Booch, 1994, p. 103) and a class is the following:

Definition 44 (Class).

A class is defined to be a set of objects sharing a common structure and common behavior.

Hence, classes are generalizing possible attributes of a set of objects, representing possible instances of objects, and can inherit their attributes to sub-classes.

Since inheritances from more than one class rise complexity of models and difficulty of interpretations, inheritances are limited to one class at the generic meta-model. This limitation can be relativized, in indicating a relationship to objects representing abstract roles Frank, 1993, p. 173.

In compliance with Frank, the clarity of models suffers from the modeling of relationships as proper objects and experiences lead to conceptual fuzzinesses (Frank, 1992; Frank and Klein, 1992). However, the here-presented object conceptualization follows Frank and considers relationships as objects as soon as attributes can be mapped to relationships (Frank, 1993, p. 195). This refers for example to the modeling of *conversions*, the *control flow*, the *control nodes*, etc.

4.1.2.2 Relationship Modeling

In general, it is useful for implementation purposes to restrict to only few kind of relations. Since the literature provides a huge amount of object-oriented modeling approaches, Frank provides a collection of 39 approaches each providing different relationships and notations (Frank, 1993, p. 114). Hence, the following chooses from available relationships from literature: associations, inheritance, aggregation, composition, using, instantiation, meta-class (Booch, 1994, p. 107; Balzert, 1999, p. 40–47).

For the design and object-oriented analysis, only two kind of relationships are required: aggregation and interaction relationships (Ferstl and Sinz, 1990; Booch, 1990, p. 88; Frank, 1993, p. 196; and Balzert, 1999, p. 40–47). While the first kind of relationship focuses on questions about the constitution via parts, components, and attributes (e.g. aggregation, composition, inheritance), the latter deals with questions about the communication of objects and classes (e.g. using). Hence, the following builds only on those essential relationships and defines for each:

- *Using (uses / used by)*: If objects are defined to have a reference to an object, they are related by a ‘use’-relation. Conditions can assure their integrity, such that for example objects are not deleted during their live time. Often, this is referred to as ‘knows’ relationship. The ‘used by’-relation is the inverse interpretation of the ‘uses’-relation.
- *Aggregation (aggregates / aggregated by)*: If objects the so-called aggregates are defined to consist of other objects the so-called parts, they are related by an ‘aggregate’-relation. Often, this is referred to as ‘has a’ relationship. The ‘aggregated by’-relation is the inverse interpretation of the ‘aggregate’-relation.
- *Composition (composes / composed by)*: If objects the so-called composites are defined to consist of other objects the so-called parts, in which the composite is responsible for the construction and destruction of the parts and parts only belong to one composite, they are related by a ‘compose’-relation. Often, this is referred to as ‘contains-a’ relationship. The ‘composed by’-relation is the inverse interpretation of the ‘compose’-relation.
- *Inheritance (generalize / specialize)*: If objects the so-called parent objects are defined to inherit attributes to further objects the so-called children, they are related by an ‘inherit’-relation. Often, this is referred to as ‘is a’ relationship. With this, hierarchies of classes can specify structural and behavioral similarities efficiently. The ‘inherited by’-relation is the inverse interpretation of the ‘inherit’-relation.

Although the differentiation of those two kinds of relations is sufficient from the viewpoint of software implementation, it is not sufficient for the purpose of modeling because a characterization is missing. Hence, each relationship is characterized by a denomination using the domain-specific terminologies. Since this denomination follows a direction, an inverse denomination can always be used alternatively. The choice of the denomination direction is selected by interpretation issues and indicates, which object knows the other object. Through this, the definition of filters allows the targeted reduction of objects visualized following a certain direction.

Beside the option of N-ary relationships, the generic meta-model is strictly limited by binary relations. Hence, only two instances of objects can be related, which supports the creation of clear models.

How many instances of objects or classes are required at least and at maximum for any relationship is specified by cardinalities. With this information, the integrity of models following the generic meta-model are ensured. Among further notation approaches, the generic meta-model follows the min-max-notation in accordance to Frank (1993, p. 196) with help of Integer intervals and literal integer values. A single asterisk indicates an unlimited range. No asterisks indicates a closed range.

4.1.2.3 Integration of Objects and Relationships

Taking the specified object conceptualization of section 4.1.2.1 and the relationship modeling of section 4.1.2.2 together, Fig. 4.3 visualizes all in one example.

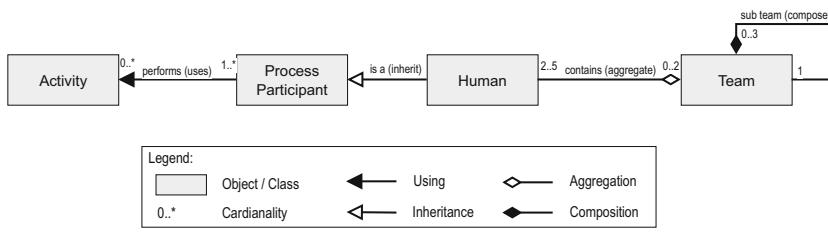


Figure 4.3 Object Model Specification

Here, one can see that an activity is performed by at least one and at max infinity process participants. A process participant can perform at least none and at max infinity activities. The corresponding kind of relationships refers to the ‘using’-relationship.

The attributes of the process participant can be found at the class of the human object because of the ‘inherit’-relationship. Hence, the human object specializes the characteristics of the process participant. Alternatively interpreted, the process participant generalizes characteristics of the human.

Humans are organized in teams of at least two and at max 5 humans. This is modeled by an ‘aggregation’-relationship. Faced with various kinds of organization structures, humans can either work independently from organizations (have no team), can work in one team or they can work in matrix organizations (have two teams).

A hierarchy of teams and sub-teams is created by a ‘composition’-relationship. While a team can have at least none and at maximum three sub-teams, a sub-team is always connected to one team.

Accordingly, when the company is ceased, the team hierarchy is ceased as well but the humans will not. This is the reason for the use of ‘composition’ and ‘aggregation’.

4.1.3 Meta-Model

The here-presented generic meta-model follows an object-oriented proceeding, as it was specified in section 4.1.2. According to a wide range of underlying concepts (see Fig. 4.2) and building blocks provided by the ANN process taxonomy (see section 4.1.1), core concepts of relevant domains have been integrated. Hence, defined perspectives, terms and semantic relations of the ANN process taxonomy flew in the meta-model, which is can be found in Fig. 4.4.

Since a description of basic objects, basic classes, and their isolated description for every perspective was part of the ANN taxonomy, here, only their integration is focused. Considering all available perspectives, it becomes clear that those are all connected with one perspective: the *Process ANN Perspective*. Hence, each connection supporting this integration, is explained in the following:

- Each *atomic neuronal activity* is performed by a *neuron*, inclusive its components and connections. Hence, the *Neuronal Perspective* is integrated with the *Process ANN Perspective*. As was described in section 2.5.1.4, the neuronal signal transmission does not differentiate the terms of *effort*, *arousal*, *hypoorousal*, *activation*, *alertness*, etc. in a first version.
- Neuronal *Network Architectures* consist of at least one *neuron*, inclusive its components and connections. Since their structural organization is intended to be interpreted within the business context, those are placed within the *Process ANN Perspective*.
- Some *neuronal activities* approximate the behavior of an individual *process participant*. Hence, the *Organizational Perspective* and the *Process ANN Perspective* are integrated.
- The performance of an *activity* is approximated by *neuronal activities*. This integrates the *Functional Perspective* with the *Process ANN Perspective*. The integration includes the *Knowledge Perspective*, the *Informational Perspective* and the *Organizational Perspective* since the performance of an *activity* consumes *resources* of the first two perspectives and *process participants* of the last perspective.
- Since *control nodes* are approximated by *neuronal activities*, the *Behavioral Perspective* is integrated with the *Process ANN Perspective*.
- The *Simulation Perspective* is integrated with the *Process ANN Perspective* as *simulation data* generated by *atomic scenarios* are mapped in *data sets* required for the *neuronal training* and *neuronal testing*.
- Business context based *goals* are reflected in the *neuronal learning context* and map into the *stopping strategy*, *learning strategy*, *test strategy* and *network*

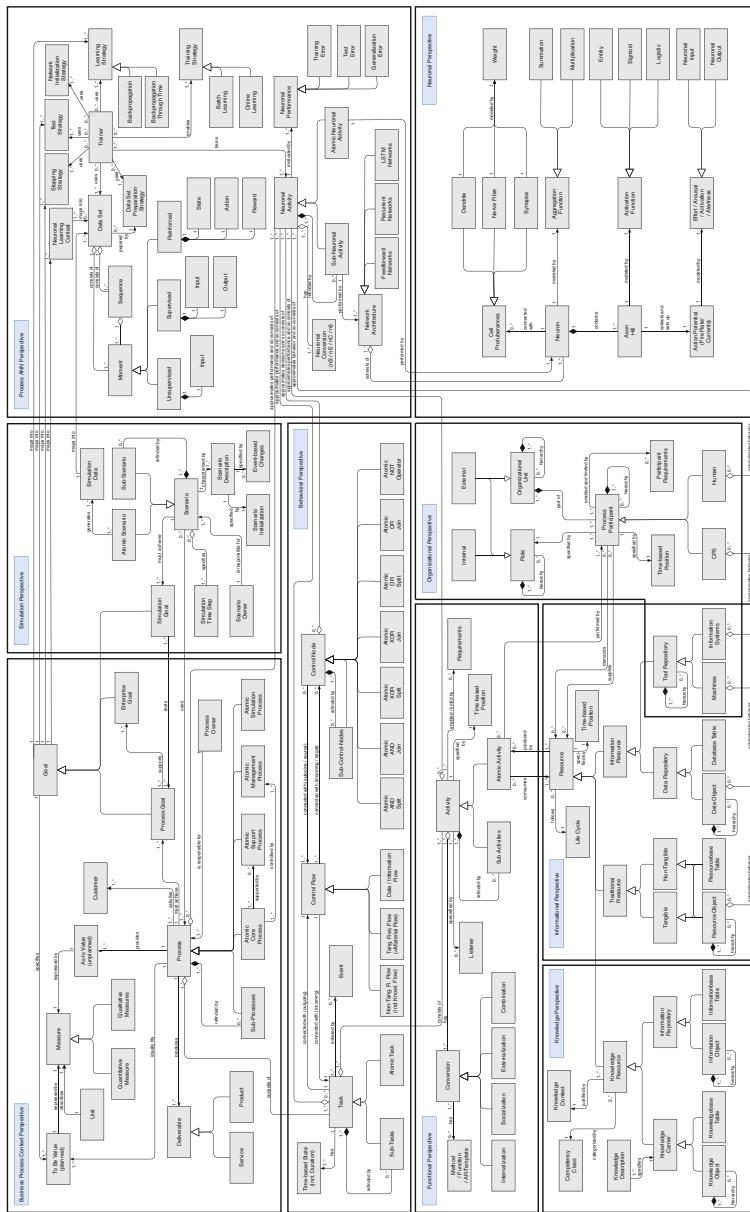


Figure 4.4 The Generic Meta-Model

initialization strategy, which integrates the *Business Process Context Perspective* and the *Process ANN Perspective*.

Modeling languages that intend to stand for a concept for neuronal modeling (CoNM) must provide model shapes for objects and classes as they are visualized with gray colored rectangles in Fig. 4.4. Hence, the here-presented object model can stand to test and review available modeling languages on the one side and available programming libraries and tools on the other.

4.1.4 Interim Conclusion

The meta-model here designed intends to stand as framework for the construction of relevant perspectives and modeling objects for a CoNM modeling language. As this represents requirements and relations about relevant modeling objects, the target-oriented integration of ANN mechanisms and process elements is enabled.

4.2 Generic Evaluation and Selection System

Following the intention to identify a technical foundation, which supports the creation of the CoNM best, first, a Generic Evaluation and Selection System (GESS) was designed, which allows to first evaluate and then select most powerful tools out of the set of tools to be analyzed. Later, it will be applied to the ANN process domain such that a Specific Evaluation and Selection System (SESS) manifests. Thus, a CoNM focus will be established.

Based on a technique-oriented understanding, the following subsumes everything as *tool* or rather *object of investigation* (OoI) that functions reliably as a means to achieve a certain goal (Fettke, Constantin, and Peter, 2010b). This includes all artificial material things as well as procedures for the application of the technique, methods, conceptual models, software, prototypes, frameworks, business models, modeling languages (Ulrich, 2010), which are applied to a certain *object of reflection* and intentionally modify this object of reflection (Wüsteneck, 1974). The described application presumes knowledge about the usefulness of certain properties of the tool, which causes a certain consequence (effect), has the same consequences when applied multiple times (repeatability), has the same consequences for the one who applies the tool (impersonality), or rather allows to reach a certain goal with better performance characteristics than without this tool (Fettke, Constantin, and Peter, 2010b).

The GESS was designed to follow an integrated view on objects of investigation. An integration considered the following three: architectural components, modeling components, and tool support. In accordance with Scheer, this is a foundation for efficient modeling concepts on the one hand and the integration must allow an evaluation of approaches not supporting all three at all on the other hand (Scheer, 2002, p. 132).

In the design of a systematic selection process, alternative forms of evaluations and variations in the consideration of severals have been considered implicitly: A focus laid on the reduction of subjective evaluations of one person, such as the author. Although an evaluation by severals and third parties increases the objectivity and subjective impressions are canceled out by great numbers of evaluators, the big picture for the construction of a CoNM is missing them. This results in qualitative lacks and an evaluation by the author is attractive. As a kind of trade-off, the following presents a selection system that is based on the evaluation of the author of this contribution first. It considers objective criteria, which are well defined and transparently demonstrated. Further, two kinds of broad evaluations by experts relativize the personal expertise of the author, so that a compromise between subjective and objective as well as quantitative and qualitative evaluations can be realized. Fig. 4.5 visualizes this.

Here, one can see that the SESS, which follows the GESS designed in the following, establishes as quantitative focus, which complements qualitative impressions of the author's evaluation following Bloom's taxonomy (Bloom et al., 1956) and being presented as part of the SLR presented within chapter 2. By this, the selection method demands for four process steps, before a selection can be carried out.

As Fig. 4.6 visualizes, specific evaluation criteria of the ANN process domain can be derived from a generic meta-model such as one has been presented in section 4.1.3.

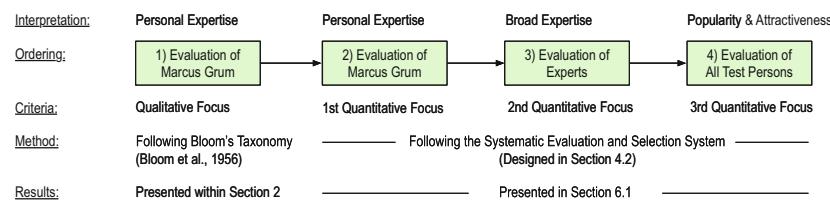


Figure 4.5 Systematic Selection Process

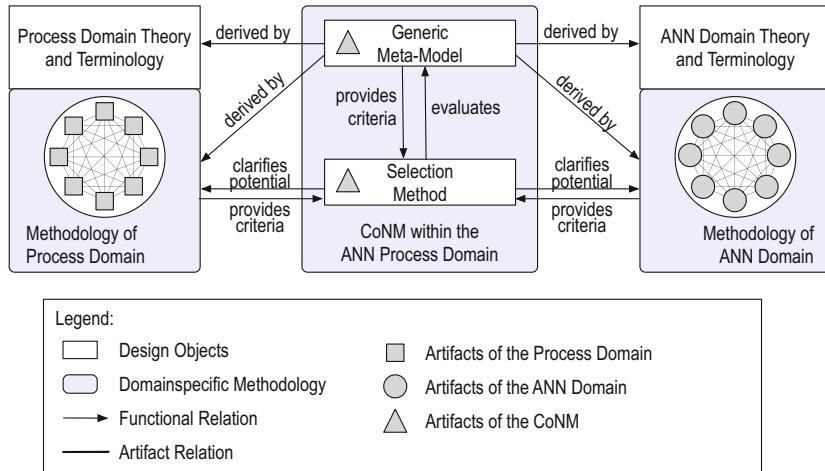


Figure 4.6 The Design of the Selection System Creation

Since here, widely accepted theory and terminology of the process domain and the ANN domain have been considered and for the CoNM required kinds of modeling objects and perspectives were identified, this supports a target-oriented, broad acceptance of concepts identified by a selection method. Further, specific criteria of modeling tools, simulation tools, and deep learning tools can be considered, which have been proven to be practical from an experience base and refer to a technical perspective. All together, those criteria help to evaluate the generic meta-model and identify the foundation for the construction of the CoNM. Further, those clarify the potential of a CoNM for the process domain and ANN domain, since a gap will become visible. In addition, when the CoNM has been constructed, the selection system can evaluate how far the CoNM fulfills the criteria. The design of the CoNM selection system is carried out in six steps, which will be presented by the following subsections.

4.2.1 Subset Definition

Imagine we have evaluation aspects $m_{i,j}$ for the construction of the CoNM, which will be concretized in the demonstration (section 6.1) and are part of the set $M_{i,j}$, those can be grouped perspective-wise by the evaluation sets P_i :

$$M_{i,j} \subseteq P_i : \Leftrightarrow \forall m_{i,j} (m_{i,j} \in M_{i,j} \rightarrow m_{i,j} \in P_i), \quad (4.1)$$

with $i, j, n_i, n_{i,j} \in \mathbb{N}$, $i = (1, \dots, n_i)$ and $j = (1, \dots, n_{i,j})$.

Further, one can assume all perspectives P_i to be part of an integrated set of all evaluation aspects called P :

$$P_i \subseteq P : \Leftrightarrow \forall m_i (m_i \in P_i \rightarrow m_i \in P), \quad (4.2)$$

with $i, n_i \in \mathbb{N}$ and $i = (1, \dots, n_i)$.

Let's further assume to have a set of objects of investigations o_k , which is called O and refers to modeling languages, modeling tools, simulation tools, and deep learning tools:

$$O = \{o_k \mid k, n_k \in \mathbb{N} \text{ and } k = (1, \dots, n_k)\}. \quad (4.3)$$

Then, we have the set of test persons t , that is referred to as T and represents experts of relevant domains and being surveyed:

$$T = \{t_l \mid l, n_l \in \mathbb{N} \text{ and } l = (1, \dots, n_l)\}. \quad (4.4)$$

This set contains the two sub-sets as follows. First, the test persons t_l^+ being able to evaluate an aspect $m_{i,j}$ of an OoI o_k , e.g. because they know the OoI to be evaluated,

$$T^+ \subseteq T : \Leftrightarrow \forall t_l^+ (t_l^+ \in T^+ \rightarrow t_l^+ \in T), \quad (4.5)$$

with $l^+, n_l^+ \in \mathbb{N}$ and $l^+ = (1, \dots, n_l^+)$

and second, the complementary test persons t_l^- being not able to evaluate an aspect $m_{i,j}$ of an OoI o_k , e.g. because they do not know the OoI to be evaluated,

$$T^- \subseteq T : \Leftrightarrow \forall t_l^- (t_l^- \in T^- \rightarrow t_l^- \in T), \quad (4.6)$$

with $l^-, n_l^- \in \mathbb{N}$ and $l^- = (1, \dots, n_l^-)$ and $n_l^- = n_l - n_l^+$.

With the abstract subset definition, the evaluation task is prepared and can be defined in a second step as follows.

4.2.2 Evaluation Task Definition

The human-based evaluation of objects of investigation by experts t_l is modeled by the function $e_l(\dots)$. Hence, the aspect-oriented valuation $v_{i,j,k}$ of an object of investigation o_k by an aspect $m_{i,j}$ is modeled by

$$v_{i,j,k,l} = e_l(m_{i,j}, o_k). \quad (4.7)$$

Here, we can establish two kinds of KPIs. First, we can define the mean μ over all test persons t_l , perspectives P_i and OoIs o_k . It represents the tendency of test persons in regard with aspect $m_{i,j}$:

$$\mu_{i,j,k} = \frac{1}{n} \sum_{l=1}^n v_{i,j,k,l}. \quad (4.8)$$

Second, we can define the corresponding variance σ^2 over all test persons t_l , perspectives P_i and OoIs o_k . It represents the tendency of test persons to deviate from the tendency in regard with aspect $m_{i,j}$:

$$\sigma_{i,j,k}^2 = \frac{1}{n} \sum_{l=1}^n (v_{i,j,k,l} - \mu_{i,j,k})^2. \quad (4.9)$$

In accordance with Fig. 4.5, these KPIs can be carried out on three bases. First, as a marginal case, focusing on the view of the author of this contribution with $n = 1$. Hence, the mean refers to the author's view and the variance equals zero.

Second, focusing on the test persons t_l^+ , that are able to evaluate an aspect $m_{i,j}$ of an OoI o_k and that are therefore part of T^+ . These are referred to as the broad expert base from hereon and $n = n_l^+$.

Third, focusing on all test persons being part of T . Since this includes both the broad expert base T^+ as well as the complementary T^- , the popularity of an OoI becomes transparent. The more experts the criteria fulfillment can evaluate because of their knowing and the better the evaluation is, the more attractive an OoI is for the CoNM foundation. Here, $n = n_l^+ + n_l^- = n_l$.

In a satisfactory manner, the functional relation of $e_l(\dots)$ can only be realized as the operationalization of all evaluation aspects is established. This will be realized in the fifth step. So far, this definition shall suffice.

Based on a perspective-wise aggregation, the valuation $V_{i,k}$ can be derived over all elements per perspective P_i by the following:

$$V_{i,k}^J = \frac{1}{n_{i,j}} \sum_{j=1}^{i,j} J_{i,j,k}. \quad (4.10)$$

Here, the J represents one of the two KPIs presented in Eq. 4.8 Eq. 4.9 as well as the base on which it is carried out. Hence, it is displayed as $V_{i,k}^{\mu,\mp}$, $V_{i,k}^{\sigma^2,\mp}$ for the author base, as $V_{i,k}^{\mu,+}$, $V_{i,k}^{\sigma^2,+}$ for the broad expert base, or as $V_{i,k}^{\mu,\pm}$, $V_{i,k}^{\sigma^2,\pm}$ for the joint test person base. The final result R_k^J can be identified over all aggregated aspects $V_{i,k}^J$ per object of investigation o_k :

$$R_k^J = \frac{1}{n_i} \sum_{i=1}^{n_i} V_{i,k}^J. \quad (4.11)$$

The final result R_k^J reflects the suitability for considered objects of investigations o_k to stand as technical foundation for the CoNM since it is an evaluation over all aspects, which focus on requirements on a CoNM. Assuming we have the vectors $\mathbf{R}^J = \{R_k^J, \forall k\}$ and $\mathbf{P}_k = \{m_{i,j,k}, \forall(i, j)\}$, then we formulate the optimization task for the construction of a CoNM as follows:

$$\begin{aligned} \max_{\mathbf{P}_k} (\mathbf{R}^J) \mid J &= \{(\mu, \mp), (\mu, +), (\mu, \pm)\}, \\ &\text{and} \\ \min_{\mathbf{P}_k} (\mathbf{R}^J) \mid J &= \{(\sigma, \mp), (\sigma, +), (\sigma, \pm)\}. \end{aligned} \quad (4.12)$$

As candidates provide low R_k^J values for $J = \mu$, they are not attractive for a foundation of the CoNM. If all candidates provide low R_k^J values or provide low values in regard to a certain aspect $m_{i,j}$, a research gap becomes visible here. Objects of investigations providing a high R_k^J (for $J = \mu$) will represent attractive foundations for the construction of the CoNM. Focusing on sigma values, low $R_k^{\sigma^2}$ values indicate a broad agreement in R_k^μ values identified. Since this indicates a clear understanding, this is desirable.

Therefore, the evaluation of the CoNM as further object of investigation is expected to outperform candidates presented here. Hence, the selection system serves for both: for the identification of foundational concepts in early research stages and the evaluation of the constructed CoNM in late research stages. This includes both, the optimization of the popularity and the technical capabilities of the OoI analyzed.

4.2.3 Visualization Design

The aggregation with help of Eq. 4.10 and Eq. 4.11 is practical since a detailed evaluation considering all 169 criteria will be presented only within the appendix - this is essential regarding transparency. Relevant aspects will be presented within the demonstration section as an overview.

All together, a schematic overview of the selection system is visualized in Fig. 4.7. Within this figure, objects of investigations are visualized column-wise and evaluation aspects row-wise, such that an evaluation matrix is created.

Its elements either refer to $v_{i,j,k}$ of Eq. 4.7 as can be seen at rows 2–5 or they refer to $V_{i,k}$ of Eq. 4.10 as other rows show. In order to create a quick graphical insight on evaluation values, a color map ranging from red to green has been chosen for the mean. In order to enable a quick overview, a second color map ranging from blue to red enables insights for the variance. Faced with exemplary valuations of

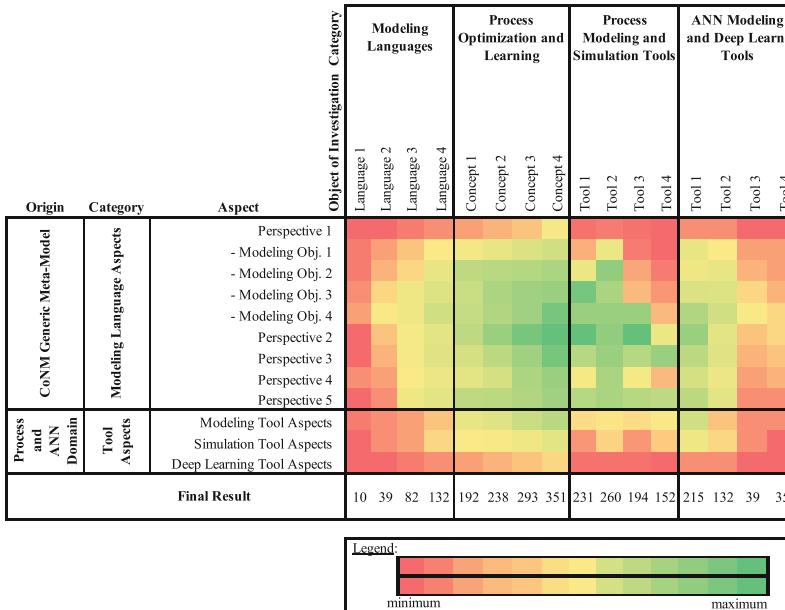


Figure 4.7 The Design of the Selection System

Fig. 4.7, one can identify attractive concepts at the center of the matrix indicated in green on behalf of the mean.

4.2.4 Evaluation Aspects Concretion

As was already explained previously with Fig. 4.6, evaluation criteria can be derived from the generic CoNM meta-models. Here, a focus lays on the availability of each kind of modeling object, which each serves as one evaluation aspect $m_{i,j}$ of Eq. 4.1.

Three further evaluation perspectives can be identified in considering relevant requirements derived from theoretical foundations for modeling and simulation tools and deep learning tools. Here, a focus lays on the provision of technical concepts, which serve each as one evaluation aspect $m_{i,j}$ of Eq. 4.1.

4.2.5 Evaluation Aspect Operationalization

Faced with evaluation aspects presented in the fourth step, the following operationalization intends to be on base of the same scale for each aspect: It focuses on the availability of each aspect in combination with the effort required for its extension given a certain object of investigation. This shall support the objectivity and guarantee a transparent evaluation proceeding. Hence, the operationalization of each aspect refers to the following scale:

- 0 points are given, if an evaluation aspect is not known.
- 1 points are given, if an evaluation aspect is neither available nor extendable at the object of investigation.
- 2 point is given, if an evaluation aspect is not available but extendable under high efforts at the object of investigation.
- 3 points are given, if an evaluation aspect is not available but a simple configuration effort is required to extend the object of investigation.
- 4 points are given, if an evaluation aspect is available and can be used as it is.

Following a set theoretic definition, imagine the scale to be part of a set called Z :

$$Z = \{z_q \mid q, z_q \in \mathbb{N} \text{ and } q = (0, \dots, 4)\}. \quad (4.13)$$

So let $D \subseteq P \times O$ be the domain D and Z be the codomain, then the function $e(\dots)$ can be defined satisfyingly to be

$$e = (G, D, Z) \text{ with } G \subseteq D \times Z. \quad (4.14)$$

Following Hamiltons definition, this means the function e from a set D to a set Z is defined by a set G of ordered pairs $(m_{i,j}, o_k, v_{i,j,k})$, such that $m_{i,j} \in P$, $o_k \in O$, $v_{i,j,k} \in Z$ and every element of P and O are the first components of exactly one ordered pair in G (Hamilton, 1983).

Since a human-based evaluation with $e_l(\dots)$ of Eq. 4.14 is operationalized only with a four point scale presented here, a detailed view of the selection system does not require a color map as it was presented at its overview in Fig. 4.7. Hence, the detailed view, which will be presented in the appendix, will use visualizations on base of quarters. This ensures a simple overview and concrete interpretation of all evaluations being created.

Further, as one separates evaluations of knowing experts and considers Eq. 4.8 as well as Eq. 4.9 for $\forall z_q \geq 1$, one focuses on a professional knowledge base. As one considers these equations for $\forall z_q \geq 0$, one focuses on the popularity and the attractiveness of the OoIs considered. Faced with the optimization task defined in Eq. 4.12, the ordinal scale presented in Eq. 4.13 enables the optimization of the OoI capabilities and their popularity and attractiveness.

4.2.6 Object of Investigation Concretion

Faced with state-of-the-art tools and concepts presented within the theoretical foundations, objects of investigation can be identified as follows:

Modeling Languages can be found in the domain of (Business) process modeling languages and knowledge modeling languages. Following the recommendation of Scheer (2002, p. 132), isolated architecture discussions without tool support or method discussions without tool support can be dismissed since an efficient practice design is not possible at all. Additionally, modeling and simulation tools can be considered since they map modeling languages and provide technical functions. Following the recommendation of Scheer, isolated graphical tools (e.g. MS Visio, OmniGraffle) that do not support architectural perspectives or modeling methods were dismissed since an efficient practice design is not possible at all (Scheer, 2002, p. 132). Further, deep learning tools can be considered as objects of investigation, since they provide occasionally graphical interfaces and deep learning techniques. Here, Scheer's recommendations for junctures of architecture, method, and tool discussions were relaxed because of deep learning tools focus on neuronal learning concepts. The OoIs will be further concretized in the demonstration (section 6.1).

4.2.7 Interim Conclusion

The selection system here designed intends to stand as framework for the evaluation of CoNM without preferring one modeling method, modeling and simulation tool and deep learning library. Since its definitions of evaluation dimensions, evaluation objects and perspectives can be modified and extended easily, changes of contemporary IT (hardware, tools, ANN techniques, etc.) can be reflected in repetitive evaluations on base of the selection system presented here. Further, the selection system can be configured easily e.g. in limiting considered modeling objects, perspectives, or evaluation objects, so that evaluations can be realized from the specific view-point of a company or model creator. This supports especially the application for small or highly specified companies.

4.3 Neuronal Enterprise Architecture

So that it becomes clear how the novel Concept of a Neuronal Modeling shall be developed in organizations and further experiments demonstrating the CoNM use are transparent in regard with their proceeding and can be reproduced easily, the following addresses the objective of having a CoNM architecture that guides the CoNM realization in organizations and its hardware and software realization (section 1.2). Although this covers a wide range of modeling areas, the management and organization of organizational objects in one organization-wide object repository is essential because the CoNM intends to approximate organizational objects and enable their reuse in overlapping process networks. Further, this is essential since the CoNM intends to realize simulations, whose real-time data streams must be interfaced from the CoNM architecture designed to CoNM models constructed and simulated. Lastly, this is essential because the CoNM intends to carry out optimizations within the CoNM models constructed and modifications need to be drawn back to the CoNM architecture.

Considering EAs available, these have been issued in section 2.1.2.3, the following artifact design establishes an environment for the neuronal modeling, which serves twofold: first, it serves to systematize the origin of neuronal activations, which function as input for neuronal processes, and second, it serves to systematize the destination for activations produced by neuronal processes, which function as corresponding output. Although the following is exemplified by the organizational context, further application fields can be addressed similarly. For this, the NEA presentation is structured as follows. First, the NEA structure is presented. Then, the associations among NEA components are issued in the second sub-section, and a

phase-wise and role-specific construction is characterized. Finally, a section-specific interim conclusion clarifies the artifacts context within this contribution.

4.3.1 NEA Structure

The Neuronal Enterprise Architecture (NEA) structure is based on the prominent and powerful EA of ARIS (section 2.1.2.3.1) and has been extended in regard with the CoNM objectives because of the following reasons: First, it is well known in science and practice. Second, it has been identified to have a powerful architecture for an organization-wide object repository in qualitative analyses. Third, it was identified as appropriate technical foundation of the CoNM in quantitative analyses, which will be presented at section 6.1. Here, the aspect-wise CoNM requirement fulfillment has been issued and the architecture mentioned has been superior in regard with corresponding aspects. Building on Def. 7, Fig. 4.8 presents data, hardware, software and communication resources as well as the supporting organization required to maintain the overall physical structure required.

The figure highlights fields of a neuronal process management and presents components required for a knowledge oriented organizational controlling. Jointly, they form the *CoNM configuration*, so that beside software components and architectural fields, a collection of components can be composed for customer-specific cases. Because of the CoNM configuration considered as framework, components can be defined to be exchangeable as so called *Hot Spots* or as not exchangeable (*Frozen Spots*), and so, the framework further refers to an incomplete application system, that adapts to a certain customer by modifications (Scheer, 2002, p. 109). The following clarifies fields of a neuronal process management and clarifies CoNM configuration components level-wise.

The first level refers to the construction of various kinds of process designs. As it holds modeling perspectives defined by the CoNM-meta-model (section 4.1), it issues on the one hand the modeling of neuronal processes within an application field, such as organizations, and on the other hand the modeling of the application field's processes, e.g. organizational processes within biologic and artificial neuronal systems. Further, it includes a *right management diagram* (Scheer, 2002, p. 112) addressing the question who is allowed to do something. Third party diagrams can be included via *model interfaces*, so that the CoNM framework is extendable by design. The *model warehouse* realizes the systematic collection, storage, and maintenance of models constructed, so that reference models can be reused and variations of models are backed up by repositories (Scheer, 2002, p. 74).

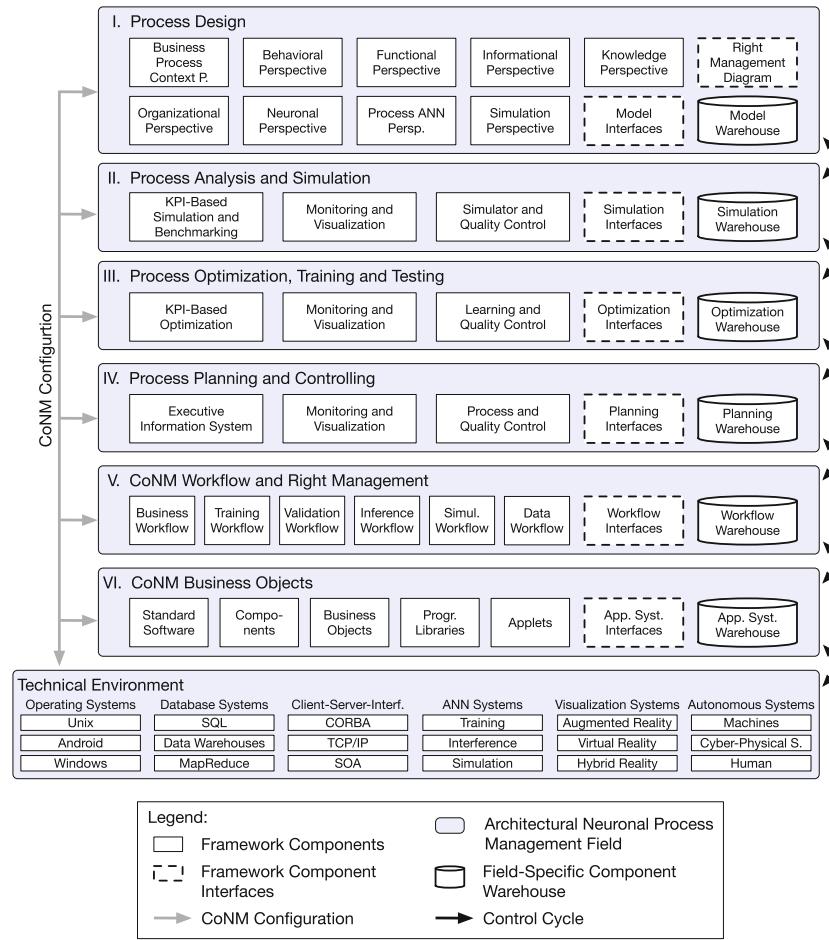


Figure 4.8 The Design of the Neuronal Enterprise Architecture Design for the CoNM

As the meta-model includes perspectives of the EA of Zachman (section 2.1.2.3.5), domains of the EA of TOGAF (section 2.1.2.3.4), and perspectives of the EA called CIMOSA (section 2.1.2.3.3) common and prominent EA perspectives have been considered by the CoNM framework.

The second level issues the use of model designs of the first level for analyses and simulation. Beyond a *KPI-based examination* and *benchmark* of (1) process variations, (2) alternatives and (3) scenarios, a *monitoring* of neuronal processes simulations (schemata and instances) as well as its transparent visualizations are realized. The *controlling* of different simulation systems and third party simulators, that have been integrated via *simulation interfaces*, is realized according to Fig. 2.36. As a proceeding for *quality assurance* is implemented, simulation runs are consistently and on a high quality base collected, stored and maintained with the aid of repositories in the *simulation warehouse*.

The third level refers to the process optimization, which includes optimization approaches from the process domain (section 2.4) as well as ANN training and testing techniques (section 2.5), that all are made applicable because of the CoNM designed. Beside the KPI-based optimization, insights in inner neuronal process structures are enabled by CoNM artifacts rising the interpretability, such as the SEKO (section 4.7), so that an adequate visualization is enabled by the CoNM modeling language. Since a proceeding for technique-specific *learning procedures* and a *quality assurance* is implemented, optimization attempts are collected, stored and maintained by repositories in the *optimization warehouse*. So, a consistent and high quality database is supported and a case-based reasoning can provide neuronal process components effectively.

The fourth level addresses the view of the *process owner*, that plans and controls ongoing processes (Scheer, 2002, p.54). With the aid of *executive information systems*, neuronal data, neuronal information and neuronal knowledge according to Def. 13 are aggregated for decisions. So, such systems support the filtering of relevant parts for the observation, supervision, analysis and planning out of a huge amount of information (Scheer, 2002, p.81). Further, a monitoring of neuronal processes (schemata and instances) as well as workflows is enabled and supported by transparent visualizations. By CoNM artifacts rising the interpretability, such as the SEKO (section 4.7), the direct controlling of processes is enabled. An adequate decision is founded on the base of neuronal simulations of the second level, so that a process planning is empowered by ANN-based predictions. Actions implemented, corresponding effects and the decision foundation are systematically collected, stored and maintained by repositories in the *planning warehouse*. It provides similar cases, so that a case-based reasoning is supported and the quality of decisions

is improved by every case considered. Further, a proceeding for *quality assurance* is implemented, so that actions are consistent and from high quality.

The fifth level focuses on the definition of workflows, that connect neuronal processes from the first level with applications and systems of the real world. So, these realize the technical routing of business objects among the neuronal control center and more or less autonomous systems. This includes the management of rights of each participant to be part of it. Here, components refer to *business workflows*, that issue the transportation of products and business objects from workplace to workplace. Further, one can find *data workflows*, that issue how digital data is routed by design from system to system. Specific workflows for the *training, validation, inference* and *simulation* of ANN systems consider the specific hardware characteristics of processing systems, current load as well as the origin and destination of requests (Grum et al., 2018). The *workflow warehouse* enables the systematical collection, storing and maintaining by repositories, so that a redundancy-free and consistent workflow data base is supported. Third party workflows can be included via *workflow interfaces*, so that the CoNM framework is extendable by design.

The sixth level refers to *application systems*, that are required for the realization of neuronal processes being defined on the first level and carried out by the fifth level. This includes the processing of business objects by the range from simple *applets* and *software components* up to complex *programming libraries* and *standard software*. As third party systems are considered via the *application system interface*, arbitrary complex *business objects* serve as input and output for neuronal processes. Because of the *application system warehouse*, any form of application system is systematically collected, stored, and maintained by repositories, so that a reusable data base is supported.

The seventh level considers the *technical environment* of the CoNM. It is so part of the CoNM configuration is extendable by any system required for the CoNM realization.

4.3.2 NEA Associations

Similar to Scheer, the levels of the Neuronal EA are associated by control cycles (Scheer, 2002, p. 55). The fourth level (*process planning and controlling*) provides information about the efficiency and profitability of neuronal processes defined in the first level (*process design*). These are the base for scenario-wise analyses in the second level (*process analyses and simulation*) as well as for process changes issued in the fourth level (*process optimization, training and testing*). If the current workload of the technical environment e.g. is to high (seventh level), workflows

of the fifth level need to change according to reasonable criteria (fourth level). Occasionally, this goes along with a change of business objects considered at the sixth level. Although Fig. 4.8 just indicates associations, the interrelation of all levels among each other becomes transparent.

4.3.3 NEA Phases and Roles

As common and prominent EA further issue a phase-wise and role-oriented construction (EA of Zachman presented in section 2.1.2.3.5 and EA called CIMOSA presented in section 2.1.2.3.3) as well the repetitive and cyclic construction (EA of TOGAF presented in section 2.1.2.3.4) and inclusion of instances (EA called CIMOSA presented in section 2.1.2.3.3), the Neuronal EA designed follows a level-wise proceeding. When a level has been finalized, the next NEA level is processed. Drawbacks are enabled at any time. Each framework component is recommended to be carried out by an individual expert. For example, we have a simulation expert for the *simulation perspective* design, an ANN expert for the *neuronal perspective* design, a knowledge management for the *knowledge perspective* design, etc.

4.3.4 Interim Conclusion

The NEA here designed intends to stand as framework for the neuronal process management. This includes a neuronal process simulation management and a neuronal optimization management. By the provision of NEA components, a systematic and target-oriented development of application systems is supported as demanded by a CoNM. Since concrete software components are provided by component-specific warehouses, the completion of an incomplete application system is simplified and the implementation of a sustainable and modifiable application for small or highly specified companies is supported.

As this model provides building blocks, it further stand as structuring method for the collection, organization, storing, and provision of tools, concepts, methods and software code supporting the CoNM realization. It will therefore guide implementations of CoNM prototypes, which will be presented in chapter 5.

4.4 CoNM Proceeding

So that it becomes clear how the novel Concept of a Neuronal Modeling shall be used and further experiments demonstrating the CoNM use are transparent in regard with their proceeding and can be reproduced easily, the following addresses the objective of having a CoNM proceeding that guides the CoNM realization (section 1.2). The proceeding for the CoNM application at any kind of enterprises is guided by the phases presented in Fig. 4.9. As the CoNM includes an AR-enabled, process-oriented knowledge management, which is extended by neuronal mechanisms, it builds on the proceeding of the KMDL presented in section 2.2.3.3. So, on the one hand, it complements phases for the neuronal modeling building on (Grum and Gronau, 2017) and on the other hand, it considers phases corresponding to the components of the Neuronal EA (presented in section 4.3). For this, the proceeding presentation is structured as follows. First, the proceeding structure is presented, which corresponds to phases. Then, the associations among phases are issued in the second sub-section. Finally, a section-specific interim conclusion clarifies the artifacts context within this contribution.

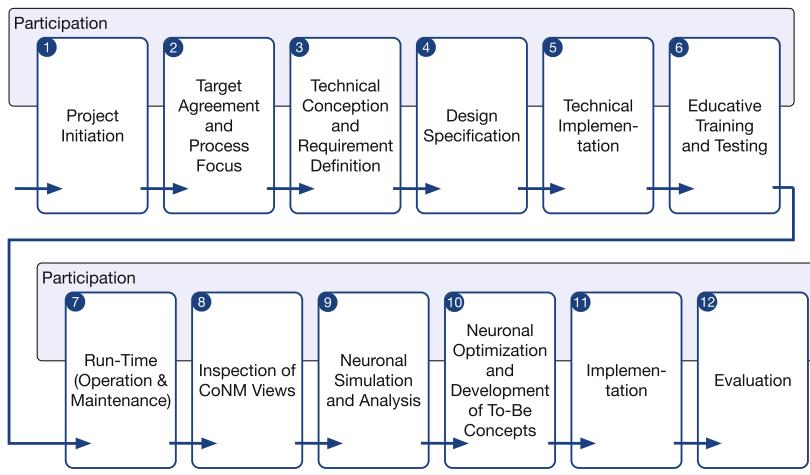


Figure 4.9 The Design of the Proceeding with the Neuronal Enterprise Architecture

4.4.1 CoNM Proceeding Phases

The following exemplifies phases according to the numbers presented in Fig. 4.9. Over the entire coarse of the proceeding, a rather collaborative than isolated proceeding is recommended, so that different kinds of stakeholder are included by participation.

- (1) The first phase refers to the acquisition of CoNM projects, and includes contractual issues, sales, and marketing, among others.
- (2) The second phase defines the focus of the learning task (Grum et al., 2020a). This includes definitions about desired temporal dynamics, learning modes (supervised, unsupervised, and reinforced learning following Deru and Ndiaye, 2019), codification schemes (following Kruse et al., 2015), learning problems (clustering, classification and regression following Nauck et al., 2013) and the intended ANN performance. Further, state-of-the-art tools as well as the company's objectives in regard with the current and planned IT infrastructure and process focus are considered as requirements (Scheer, 2002, p. 39). A target-oriented CoNM system specification is supported by KPIs the CoNM system must satisfy.
- (3) Within the third phase, the modeling of knowledge-intensive processes is realized, which will serve as foundation for the definition of learning tasks from a transparent and intuitive perspective of non-ANN experts. Here, processes are selected, which are relevant for the specification of the learning task intended and requirements are defined for Neuronal EA components. Abstract and high-level CoNM models serve as a technical conception in the sense of Scheer's second step ("requirement definition"), so that formalized description languages can serve as starting point for a technical implementation (Scheer, 2002, p. 40).
- (4) In the fourth phase, the knowledge generation is specified. This resembles the Scheer's third step called "design specification" (Scheer, 2002, p. 41). The focus issued here lies on the detailed and granular specification of CoNM models and Neuronal EA components, so that a technical implementation of the ANN environment is prepared on behalf of models constructed. Questions about knowledge carriers, the identification of information systems as knowledge bases and the characterization of concrete knowledge objects are issued. Further, the knowledge use is specified, so that it becomes clear how ANN-based results are considered for the realization of processes, such as business processes or neuronal processes.

-
- (5) In a fifth phase, the learning task can be prepared with the aid of models constructed so far. As a technical implementation (fourth step of Scheer, Scheer, 2002, p. 41), Neuronal EA components are prepared. Examples can be found in the workflow implementation, automated data collection and set realization, a rough data cleaning and transformation, etc. In general, any activity is carried out, so that a learning can be realized efficiently in the next phase.
 - (6) In a sixth phase, the ANN systems designed will be trained, tested, and evaluated on behalf a method for the development of neuronal networks (e.g. section 3.2.3), up to the point the AI satisfies the performance agreed with in the second phase. As this phase probably demands for specialized proceedings supporting individual ANN technique aspects, specialized proceedings can be nested here in the sense of a mixed method (Cresswell, 2014; Kuckartz, 2014). Since ANN systems consider as-is data, the **Neuronal Process Modeling Circle (NPMC)** is satisfied according to Fig 4.10 (a). Here, on behalf of cyclic phases, models are constructed iteratively, which is similar to CIP as follows: In the *plan phase*, relevant data input and output is planned. Then the neuronal activation is generated (*do phase*). Third, in the *check phase*, an error-similar construct is determined, so that ANN structure modifications can be carried out during the *act phase*. CoNM tools, such as the learning principle of neuronal recursion designed in section 4.8.1, support the iterative model improvement as interpretable structures are kept, while the performance is improved.
 - (7) In a seventh phase, the AI system is continuously run, maintained and improved in the sense of CIP (section 2.4.2.2). This issues the optimization of CoNM tools, dashboards and the efficient CoNM realization. So, the functioning of the CoNM system incl. its Neuronal EA components (presented in section 4.3) is guaranteed. Further, the operational CoNM system use as decision support system as well as neuronal control center is realized. More strategic issues focusing on CoNM-based changes, are addressed by later phases.
 - (8) The eighth phase addresses the inspection of CoNM views constructed by both, human-based as well as machine-based model constructors. This includes activities of a Neuronal Knowledge Management (section 4.5), that are not only performed by humans, and deals with models created on behalf of the mechanism called Systematic Exhaustion of Knowledge Objects (section 4.7). As models being constructed consider as-is data and the weight adjustments are based on as-is data, the NPMC is satisfied according to Fig 4.10 (a).
 - (9) The ninth phase considers ANN models constructed for simulation purposes. Here, different ANN versions and simulation scenarios are carried out, so

that an iterative and cyclic proceeding results in model adjustments similar to CIP as follows: In the *plan phase*, relevant as-is and to-be scenarios are planned. Then the neuronal activation is generated by the use of ANN models constructed (*do phase*). Third, in the *check phase*, an error-similar construct is determined by the comparison of the neuronal behavior and initial expectations. Modifications can be carried out during the *act phase*, so that best models for relevant scenarios are identified for this iteration. As models being constructed consider as-is simulation data and to-be simulation data, the **Neuronal Process Simulaton Circle (NPSC)** is satisfied in accordance with Fig 4.10 (b). CoNM tools, such as a spiral of neuronal knowledge creation, which applies Nonaka and Takeuchi's knowledge spiral on a neuronal level (section 2.19), support the iterative analysis and simulation of knowledge distribution over the whole organization and each ANN system.

- (10) By the tenth phase, the neuronal optimization is issued. This is either affected by the **Neuronal Knowledge Management** (section 4.5), or by the learning principles of organizations and ANN. As models being constructed consider as-is input data and the weight adjustments are based on to-be data, the **Neuronal Process Optimization Circle (NPOC)** is satisfied according to Fig 4.10 (c). Since optimizations are addressed within the model system environment (Fig. 2.1), for the derivation of to-be concepts, the relations to the original system must be established, so that implementation strategies issue the transformation of the original system.
- (11) Then, in the eleventh phase, any kind of change is implemented within the original system (Fig. 2.1). This can refer to changes in physical objects, organization principles applied in reality, to any kind of flows considered (e.g. material flows, data flows, knowledge flows, behavior-induced flows), to changes in data, information and knowledge bases of machine and human process participants, etc. Faced with the NEA designed (section 4.3), the ANN systems hold the potential to function as organizational control center when integrated with workflow systems that automatically realize decisions by the ANN systems.
- (12) As any kind of change is implemented in the original system, the twelfth phase addresses the examination of benefits achieved by changes. When appropriate, changes are observed within the original environment for a certain period of time. Furthermore, since changes have not improved the original system, and do not support the achievement of objectives agreed in the second phase, changes need to be rolled back.

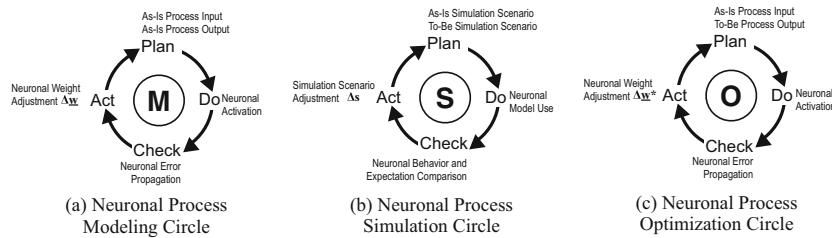


Figure 4.10 The Neuronal Process Circles

4.4.2 CoNM Proceeding Associations

Although Fig. 4.9 and Fig. 4.10 imply that the next phase is processed, when a phase has been finalized, drawbacks are enabled at any time. So, e.g. model adjustments because of the NPOC (tenth phase) need to be reflected by the workflow maintenance of the seventh phase. Further, each phase is recommended to be carried out by an individual expert. For example, we have an expert for contractual issues during the first phase, workflow management and third party programming experts for the technical implementation of data flows and third party product integration, and an ANN expert for the sixth phase, etc. In any case a model is constructed, a validation needs to be carried out in cooperation with the customer (Grum and Gronau, 2017). Further, if changes are suggested, the potential can be evaluated in workshops with the customer (Grum and Gronau, 2017).

4.4.3 Interim Conclusion

The CoNM proceeding here designed intends to stand as framework for the project-based application of the CoNM. This refers to the more or less extensive implementation of an application system in various fields of applications. Examples can be found in CoNM application systems representing business processes of companies, quantum processes measured by physical instruments, reaction processes of chemical industries, etc.

As the CoNM proceeding will methodologically guide the demonstrations, that address the CoNM functioning and are presented in chapter 6.

4.5 Neuronal Knowledge Management Model

Having identified in qualitative analyses of chapter 2 and its subsections, as well as in the SLR (section 2.7), that not any contemporary KM (cf. section 2.2.2) model considers neuronal perspectives and not any approach to deal with ANN (cf. section 2.5) or tool to deal with ANN (cf. section 2.6.2) considers traditional KM activities, the following establishes a Neuronal Knowledge Management (NKM). Since the Potsdam KM Model has been identified as powerful representative to carry out a KM (cf. section 2.2.2.3), this model has served as starting point for an extension, which considers the transfer to the neuronal context.

So, in this section, a model is designed issuing the ANN Process Domain. It clarifies how the novel CoNM (Concept of a Neuronal Modeling) shall be applied in organizations in order to symbiotically deal with human as well as artificial knowledge carriers. Further, it clarifies the CoNM use in experiments because measure-based management interventions or rather modeling activities can be made transparent and can experiments can be reproduced easily. Hence, the following addresses the objective of having a measure-based management model that guides the symbiotic CoNM realization with the aid of human as well as artificial knowledge carriers (section 1.2). From hereon, this is called Neuronal Knowledge Management Model (NKMM), which builds on the following three: first, a KM model, which addresses the process domain and refers to the Potsdam KM Model; second, a NKM model, which addresses the neuronal domain and is a new model; third, a management model addressing activities from the one domain in the other domain and vice versa. This guarantees that common KM measures and activities are realized aside the new kind of NKM measures and activities demanded by the CoNM.

Preliminary NKMM versions have been discussed by Grum (2020) and Fig. 4.11 presents the final NKMM as overview. Since new components have been highlighted in gray, the integration of classical KM components and new components of the NKM model in the NKMM become transparent. As basic KM components already have been explained in section 2.2.2.3, the following focuses on extensions by a NKM and issues changes, that occur because of the integration mentioned.

The NKMM's *design objective* refers to the creation of an ordering system, that structures KM and NKM activities jointly (*ordering amounts*), so that an adequate management of processes and knowledge carriers, which is in the sense of the CoNM, can be carried out. The concrete distinction of the NKMM components has been visualized in Fig. 4.12.

Here, classical KM components have been indicated by highlighting the bottom left corner. The new NKM components have been indicated by highlighting the top right corner. The joint KM and NKM components of the NKMM have been indicated

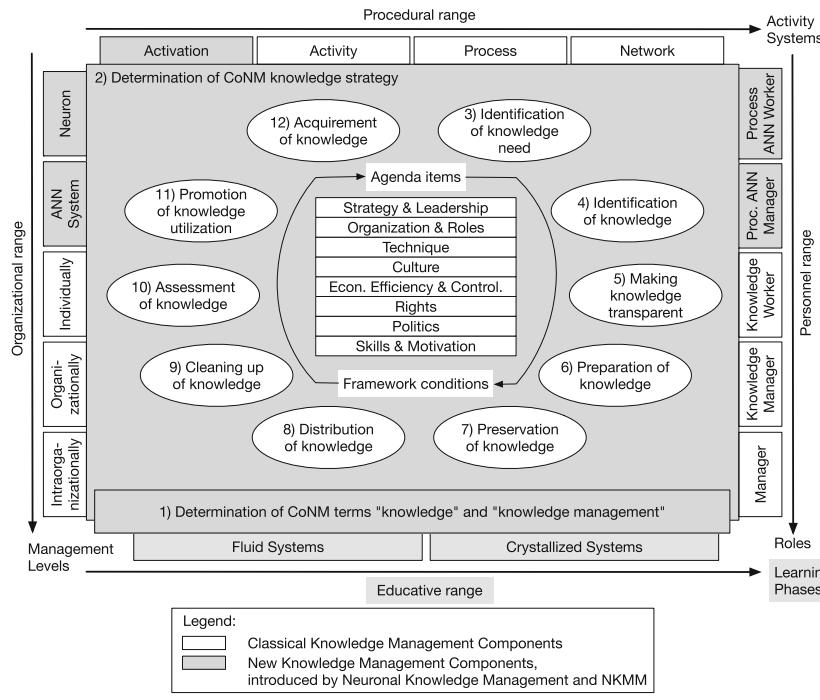


Figure 4.11 The New Components of the Neuronal Knowledge Management Model following Grum (2020)

by a gray shading. Hence, by the three kinds of visualizations, the integration of three individual models to manage knowledge jointly becomes transparent.

The *ordering characteristics* follow the three dimensions of the Potsdam KM Model (Gronau, 2009, p. 49) as follows: The NKMM and the NKM model provide various forms of activity systems for the *procedural range*, various forms of management levels for the *organizational range* and various forms of roles for the *personnel range*. Further, on base of pedagogical and cognitive-psychological perspectives, various phases are supplemented for the *educative range*. Each is exemplified in the following subsections. The dimensions' *manifestations* and *ordering principles* will be presented in the following subsections individually.

4.5.1 Activity Collection

Beside the KM activities that have been described by the Potsdam KM Model (section 2.2.2.3; Gronau, 2009, p. 49), the NKMM and the NKM model consider the same activities within the ANN domain. Jointly, they are referred to as *ordering amounts*. For the new KM models (NKMM and NKM model), the range of KM activities has not been modified but extended by NKM activities having the same naming of the Potsdam KM model, but they follow an interpretation within the neuronal context. Hence, the following provides new NKM activities by reinterpretation of KM activities. The numbers visualized in Fig. 4.12 refer to the following explanations and do not represent a sequential order.

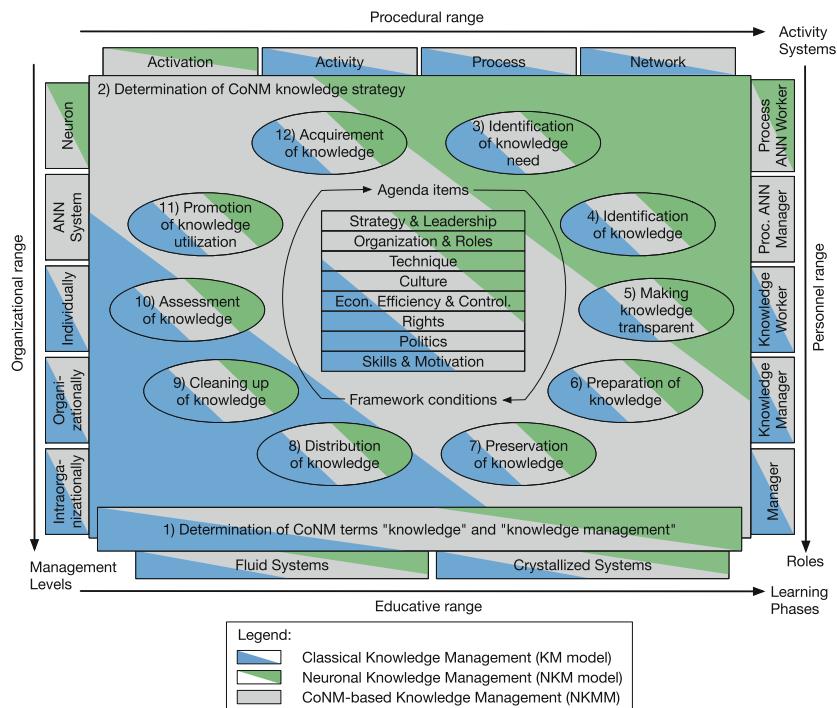


Figure 4.12 The Design of the Neuronal Knowledge Management Model

1. *Determination of terms:* A common understanding of relevant terms is established so that the whole organization is able to communicate efficiently. Unlike the Potsdam KM Model, this refers to the CoNM terms of “knowledge” and “knowledge management”, which reflect the ANN process domain and builds on the CoNM designed in this contribution.
2. *Determination of knowledge strategy:* Being part of the organizational strategy, the importance of knowledge is determined from both, the KM and NKM perspective. This includes the establishment of an estimation about the adequate effort to obtain and extend the organizational knowledge base and the neuronal knowledge base regarding financial, technical and organizational dimensions and the determination of strategic KM and NKM objectives. On base of this strategy, the subsequent ten operational KM and NKM tasks can be derived and concrete measures are mapped to strategic KM and NKM objectives.
3. *Identification of knowledge need:* Beside the identification of KM-based knowledge needs issued by the Potsdam KM Model, the NKM model examines on an individual level of a group of neurons, which kind of knowledge is required for an effective process step realization and knowledge generation. This includes the characterization of ANN-based knowledge by a description, time point and location, so that temporal and spatial circumstances are considered. Further, a description on behalf of competences (educational understanding, cf. section 2.2.1.1.3) and context information (KM- and KL-oriented understanding, cf. section 2.2.1.1.5) is included. The NKMM further considers ANN systems, their subsystems as well as their neurons. Measures here issue the need of knowledge from the process domain in ANN systems as well as ANN-based knowledge needs in the process domain. Since the need is regarded in compliance with the systems role within the network of systems, this might be inspired by pedagogical and cognitive-psychological types of systems (cf. section 2.2.1.1.4) as well as system-specific capacities and burdens according to Fig. 2.18, measures and activities intend to support the successful construction of schemata (the first phase of learning).
4. *Identification of knowledge:* Beside the recognition of the personal, organizational and inter-organizational knowledge base issued by the Potsdam KM Model, the current knowledge base of the ANN system is examined by the NKMM in regard with the currently available knowledge. This includes ANN-based knowledge representations on a personal, organizational and inter-organizational level. The NKM model here focuses on the group of neurons only. A focus lays on the differentiation of tacit and explicit knowledge, information and data, so that an adequate possibility to identify these entities is issued and the tacit dimension of knowledge is considered (KM- and

KL-oriented understanding, cf. section 2.2.1.1.5). Here, a hierarchical and taxonomic structure is supported, so that the construction of complex invariants, schemata and patterns is considered. By this, the neuro-scientific and cognitive-scientific understanding (cf. section 2.2.1.1.7) and pedagogical and cognitive-psychological understanding (cf. section 2.2.1.1.4) is regarded, which is subsumed as knowledge (Grum and Gronau, 2018a).

5. *Making knowledge transparent:* Beside the identification and visualization of knowledge of internal and external knowledge carriers (knowledge workers) in an appropriate manner issued by the Potsdam KM Model, the NKM model identifies knowledge on a neuronal level. Going beyond, the NKMM identifies knowledge of any kind of neuronal level, makes it transparent by visualizations and supports its access by meta-information for ANN systems and human knowledge carriers in an appropriate manner. By this, the existence and location of knowledge currently available on a neuronal level becomes more transparent and an interpretation is enabled. Here, the selecting, comparing, evaluating, and inferring as a plausible act of dealing with knowledge of systems is considered (cf. section 2.2.1.1.7). Since meta-information include the characterization of its situation, its problem to solve and its organizational objectives to meet, the contextual meaning is recognized (KM- and KL-oriented understanding, cf. section 2.2.1.1.5). As a denomination of human workers is included, this can stand aside a machine-based denomination, the knowledge understanding without a clear meaning (everyday-based understanding, cf. section 2.2.1.1.1) is supported, and an understanding on base of family resemblances is considered.
6. *Preparation of knowledge:* In harmony with the Potsdam KM Model, the NKMM issues the idea that knowledge is transformed to a KM-friendly and a NKM-friendly form, which allows the efficient perception and use in current and future task realizations. This includes neuronal tasks, activities, processes, and business processes, e.g. In distinction of the NKMM, the NKM model transforms knowledge to a NKM-friendly form only, which excludes the KM-friendly form that enables an integrative management of KM and NKM measures as well as its activities. Here, the design of adequate chunks is issued, so that individual capacities of learning systems and individual burden are considered (pedagogical and cognitive-psychological knowledge understanding, cf. section 2.2.1.1.4).
7. *Preservation of knowledge:* Beside the structuring, decontextualization and backing up of knowledge from the KM perspective issued by the Potsdam KM Model, the NKMM and the NKM model, both consider the structuring and backing up of ANN-based knowledge, so that the efficient provision of knowledge for any other ANN system is supported. Because of the use of

context-based knowledge in other contexts, this is connected with a knowledge decontextualization from the initial context as well as a correct generalization. In distinction to the NKM model, the NKMM harmonizes the structuring and backing up of knowledge of ANN systems and organizational entities, so that an effective exchange is supported. While some systems function as short-term memory (fluid system) and some function as long-term memory (crystallized system), measures and activities focus on the stabilizing and sustainable anchoring of schemata built (pedagogical and cognitive-psychological knowledge understanding, the second phase of learning, cf. section 2.2.1.1.4).

8. *Distribution of knowledge:* A knowledge distribution within the organization must not be limited with respect to by management or by culture induced knowledge distributions, as the Potsdam KM Models issues it. In addition, the NKMM as well as the NKM model issue the distribution of knowledge within ANN systems, regarding the efficient allocation for neurons having a certain knowledge need. Here, one can find direct management measures or self-organized distribution, such as by adequate algorithms e.g. The NKMM further supports the distribution of ANN systems within the organization and the distribution of knowledge of the organization within ANN systems. Here, measures and activities address the model of information sender and information receiver being connected by channels and exchanging messages (IS-oriented and AI-based knowledge understanding, cf. section 2.2.1.1.6). So, messages can refer to both, an objective truth and approximative values.
9. *Cleaning up of knowledge:* Beyond the correction, renewing, and deletion of knowledge from the KM perspective issued by the Potsdam KM Model, the NKM model cares about the correction, renewing and deletion of ANN-based knowledge, so that negative effects of faulty knowledge or knowledge, that will not be demanded by the organization any more, can be avoided. This supports the efficient use of ANN resources and considers the following two: first, the cleaning of static knowledge (e.g. neurons, weights, transmitter), and second, the cleaning of dynamic knowledge (e.g. activations). By this, the neuro-scientific and cognitive-scientific understanding is regarded (cf. section 2.2.1.1.7). Since the NKMM additionally issues the interplay of knowledge from ANN systems and organizations, negative effects are minimized because of knowledge of the one knowledge base on effects of the other domain.
10. *Assessment of knowledge:* The assessment of the current knowledge base of an organization in regard with its value within the entire value chain issued by the Potsdam KM Model. The NKMM additionally includes the assessment of the knowledge base generated by ANN systems. With this, it is assured that the objectives of the KM and NKM strategy are met. The NKM model

limits the knowledge assessment on neuronal systems only. Hence, only the fulfillment of NKM objectives can be issued, here (IS-oriented and AI-based knowledge understanding, cf. section 2.2.1.1.6). Since the assessment builds on meta-information identified, and recognizes if organizational objectives have been met, the contextual meaning of knowledge is recognized (KM- and KL-oriented understanding, cf. section 2.2.1.1.5).

11. *Promotion of knowledge utilization:* In addition to the creation of new organizational structures, processes and systems or the modification of existing ones for the support of a knowledge use, which is established by the Potsdam KM Model, the NKMM includes the creation of new ANN structures, systems, subsystems and neurons or the modification of existing ones, so that ANN-based knowledge is used throughout any kind of networking systems. Further, the creation of an ANN-friendly climate within the organization is issued, so that ANN-based results are accepted. The NKM model here only focuses on the constructions of neuronal structures in regard with the ANN performance. Here, analogies from ANN systems and organizational entities are disregarded.
12. *Acquirement of knowledge:* Unlike the elimination of a knowledge deficit by external acquisition or education, this is issued by the knowledge management perspective of the Potsdam KM Model, the ANN knowledge deficit is treated by the NKM model as follows: If a knowledge deficit is recognized within the ANN system, it is overcome by one of the following examples for knowledge acquisition: (1) the external acquisition of educated ANN subsystems from the model repository, (2) the inclusion and training of further subsystems designed anew, so that more learning capacity will overcome deficits, (3) the modification of the current ANN system, so that knowledge blockades are erased, (4) the training prolongation, so that relevant knowledge is generated, or (5) the target-oriented training variation, so that more global learning optima are achieved. The latter can e.g. refer to the inclusion of further training data, a changed knowledge codification, different training algorithms. Since the NKMM includes the KM and NKM, the knowledge acquisition by education or external parties is considered for ANN systems, as well as ANN systems are considered for the elimination of a knowledge deficit within an organization. Since the acquirement of knowledge can be associated with competences, the learning process e.g. can be operationalized competence levels by Bloom et al. (educational understanding, cf. section 2.2.1.1.3), the educative knowledge understanding is considered.

4.5.2 Educative Range Specification

The educative range focuses on the question, in which phase the learning system currently is, so that KM and NKM measures and its concrete KM and NKM activities can consider the type of the system (its role within the network of systems), its individual capacities and learning burdens as well as its current knowledge state for the realization of an educative, systematic, target-oriented development of learning systems. Its manifestations are ordered by the learning progress of management levels being focused by a certain KM and NKM measure. Since an educative proceeding has not been considered by any KM model, yet, and all activities are affected by the educative range, these activities have been presented in section 4.5.1, the following issues changes for the KM model, NKM model and NKMM.

1. *Fluid Systems:* During the first learning phase, capacities are addressed by measures and activities, that reduce learning barriers and enable the schemata building in fluid systems. While KM measures and activities address capacities of traditional knowledge workers and human process participants, NKM measures and activities address capacities of ANN systems and machine-based process participants. So, by following the overload theory (Fig. 2.18), a subjective difficulty is reduced by the acquisition of prior knowledge, relevant for the task realization, and individual factors, such as worries, fears, priorities, etc. when available. By this, the individual cognitive load is reduced. Objective difficulties are addressed by the system specific chunk design. So, the intrinsic cognitive load of a system is managed. This can refer to the definition of training tasks for ANN, as well as workshops for human participants. A pedagogical design of learning tasks intends to reduce an extraneous cognitive load. The Germane cognitive load is issued by measures and activities addressing the optimal resource provision. In the case of ANN systems, this e.g. refers to the provision of processing power, storage and learning time. In the case of human-based management levels, this e.g. refers to the provision of training time, and productive environmental conditions. The NKMM jointly considers the interplay of human and ANN-based learning systems, so that a cooperative dealing with learning tasks is enabled and knowledge can be build as schemata in all kinds of management levels.
2. *Crystallized Systems:* During the second learning phase, free capacities for the storing of schemata are addressed, so that knowledge is integrated with and anchored in the crystallized knowledge base sustainably. Primarily, KM measures and activities address schemata of traditional knowledge workers and human process participants, that have been built during the first learning phase.

Corresponding NKM measures and activities address fluid schemata of ANN systems and machine-based process participants. The NKMM jointly considers the sustainable knowledge anchoring of any kind of fluid systems in any kind of fluid system.

Although various phases are adequate for an educative development of learning systems, as a rough first level, pedagogical and cognitive-psychological types of systems and corresponding learning phases, capacities, and burdens have been considered (cf. section 2.2.1.1.4). Further possibilities refer to e.g. (1) the refinement of phases for different kinds of systems, such as phonological loop, episodic buffer, visuo-spatial sketch pad, central executive, (2) the six competence levels of Bloom, such as *knowing, comprehending, applying, synthesizing, analyzing and evaluating* (Bloom et al., 1956), each higher level demanding for more complex and effortful mental processes (educational knowledge understanding, cf. section 2.2.1.1.3), (3) a phase-wise condition implementation as opinion building process (philosophical knowledge understanding, cf. section 2.2.1.1.2), (4) ANN learning procedures (cf. section 3.2.3). The essential point of this range rather refers to the systematic education of learning systems than to the concrete design.

4.5.3 Organizational Range Specification

The organizational range focuses on the question of which areas of an organizational structure are affected by a KM measure and its concrete KM activities (Gronau, 2009, p. 49). Its manifestations are ordered by the amount and size of organizational structures participating in a certain KM measure. While the NKM model only considers the level of *Neurons* to be affected by NKM measures and NKM activities, for the NKMM, the Potsdam KM Model has been extended by the elements of *ANN Systems* and *Neurons*, so that a NKM is enabled and the following issues what changes occur.

1. *Neuronal*: The neuronal manifestation issues the very atomic knowledge carriers as management level. It refers to compartments that jointly construct individual ANN systems. These are substantiated by generic mechanisms so that each component of the neuronal management level functions as atomic management component. While the NKM model considers this level only for the application of NKM measures, the NKMM intends to project KM measures and activities on the neuronal level as well as to project NKM measures and activities on the activity level, process level and network level. A classical KM model so far

- has not accessed this kind of management level because of the focus on KM measures and activities.
2. *ANN System*: As ANN systems substantiate higher management levels, such as the *individual*, the own *organization* in whatever detail (team, department, section, division, etc), as well as *external organizations* by ANN mechanisms and algorithmic representations, the manifestation of ANN systems issues KM and NKM measures and activities that affect neuronal representations. Since neither the NKM model nor the Potsdam KM model consider systems from the other domain, this level functions as bridge between different kinds of systems.
 3. *Individual*: Beside the idea that KM measures and activities affect individuals, such as roles like the knowledge worker or persons like Max Mustermann, issued by the Potsdam KM Model, the NKMM includes the idea that NKM measures and activities can affect individuals, too. Examples can be found at knowledge generated by ANN systems, which is considered by process participants in the sense of decision support systems, or a changed task realization sequence that follows biologic principles. The NKM model is designed to disregard this kind of management level, since NKM measures and activities focus on the efficient ML task realization.
 4. *Organizational*: In addition to KM measures and activities affecting organizational entities, such as groups, teams, departments, and sections, issued by the Potsdam KM Model, the NKMM includes the idea that NKM measures and activities can affect these entities, too. Examples can refer to the team combination following a neuronal structure, or a department creation, that follows the cooperative behavior of neuronal dynamics. The NKM model does not consider this kind of management level, since NKM measures and activities focus on the efficient ML task realization.
 5. *Intraorganizational*: The intraorganizational manifestation of the Potsdam KM Model addresses the management level of organizations. This addresses KM measures and activities, that affect the collaboration between enterprises, the relation to suppliers and external service providers, e.g. In addition, the NKMM reflects NKM measures and activities for the affecting of intraorganizational entities. For example, one can find competitive ANN systems representing competitors. Having a clear focus on the ML task realization, the NKM model does not consider this kind of management level.

Since the dealing with knowledge is recognized throughout all management levels presented and the interplay of these levels is enabled because of a joint knowledge definition (Def. 13), the social construction of knowledge is considered (KM- and KL-oriented understanding, cf. section 2.2.1.1.5).

4.5.4 Procedural Range Specification

The procedural range focuses on the question of which areas of a process organization are affected by a KM measure and its concrete KM activities (Gronau, 2009, p. 49). Its manifestations are ordered by a kind of complexity and complicatedness of manifestations participating in a certain KM measure. The Potsdam KM Model has been extended by the element of addressing neuronal activities, the so called *Activation*, so that a NKM is enabled by neuronal activities, which are relevant for a certain behavior of an ANN system. The following issues what changes occur.

1. *Activation*: As the key activity demanded by the CoNM meta-model (designed in section 4.1 and summarized by Fig. 4.4), the neuronal activity brings together ANN mechanisms from the ANN domain and a process understanding of the process domain. It refers to the most atomic activity system available of all and translates measures of a KM as well as a NKM to the neuronal context. Since measures or activities of KM and NKM can affect neuronal activities and corresponding ANN representations, the NKMM addresses an integrative activation understanding, so that neuronal activities are addressed being relevant for the behavior of an ANN system, which is in the sense of the CoNM. While the NKM model considers this level only, a classical KM model so far has not accessed this kind of activity system.
2. *Activity*: The activity manifestation of the Potsdam KM Model addresses the idea that a measure or activity of KM can affect an individual activity. By the NKMM, additionally, NKM-based measures and activities are assumed to affect individual activities, too, and the NKMM addresses an integrative activity understanding in the sense of the CoNM: first, measures or activities of KM can affect corresponding ANN representations, second, measures or activities of NKM can affect activities of process participants and knowledge carriers. The NKM model is designed to not consider this kind of activity system, since NKM measures and activities focus on the efficient ML task realization.
3. *Process*: The process manifestation of the Potsdam KM Model issues the activity system level of sequential processes, such as a simple, value-added process of the product development (Gronau, 2009, p. 50). Since measures or activities of NKM are considered to affect entire processes, too, and measures or activities of KM can affect corresponding ANN representations, the NKMM addresses an integrative process understanding in the sense of the CoNM. The NKM model is designed to disregard this kind of activity system, since NKM measures and activities focus on the efficient ML task realization.

4. *Network*: The network manifestation of the Potsdam KM Model issues the most complex and complicated activity system level of non-sequential processes, that must be considered as a mesh of processes. In addition, the NKMM considers measures or activities of NKM to affect entire networks, too, and measures or activities of KM to affect corresponding ANN representations. Hence, the NKMM addresses an integrative network understanding in the sense of the CoNM. Having a clear focus of NKM measures and activities, the NKM model disregards this kind of activity system.

4.5.5 Personnel Range Specification

The personnel range focuses on the question, which persons are part of the realization of a KM measure and carry out concrete KM activities (Gronau, 2009, p. 49). Its manifestations are ordered by a kind of professional hierarchy. The Potsdam KM Model has been extended by the element of *Process ANN Manager* and *Scientific Expert*, so that a NKM is enabled and the following issues what changes occur.

1. *Manager*: Beside the persons caring about the efficient realization of a model, this is issued by the Potsdam KM Model, because of the NKMM, the manager will be faced with the efficient and economic realization of CoNM-based models. Focusing on the NKM model only, this does not consider the management role directly, because the effect of a manager is decoupled from the technical ML task realization. It rather must be seen as environmental conditions for technical experts.
2. *Knowledge Manager*: The knowledge manager cares about the efficient dealing with the resource of knowledge within the company (Probst, Raub, and Romhardt, 2006). According to the Potsdam KM Model, this is carried out on behalf of contemporary KM approaches. As the NKMM establishes a NKM in addition, the knowledge manager includes knowledge generated by ANN systems, subsystems and neurons so that the organizational knowledge base is supplemented. Since knowledge managers are not considered for the technical ML task realization, these rather serve as domain-specific experts for the design, the NKM model does not consider this kind of role directly.
3. *Knowledge Worker*: According to the Potsdam KM Model, the process participant is referred to as knowledge worker, that carries out knowledge-intensive processes. In addition, in the NKMM, it will deal with the ANN constructed and stands as foundation for ANN-based representations. Hence, beside of human workers, autonomous systems and any kind of object being managed can be

considered as knowledge worker. Knowledge workers are not considered by the NKM model directly, since these rather function as domain-specific experts for the design of ANN systems.

4. *Process ANN Manager:* This role must be seen as counterpart for the knowledge manager. The process ANN manager cares about the efficient dealing with the resource of knowledge within a company on behalf of contemporary CoNM approaches. In this person, we can find the CoNM expert focusing on a wide range of CoNM tools and a broad understanding of knowledge. Since neither the NKM model nor the Potsdam KM model consider systems from the other domain, this role functions as bridge between different kinds of systems.
5. *Process ANN Worker:* As a kind of complement for the process ANN manager, the process ANN worker issues the very specific competence of an scientific expert. It provides expertise from at least one domain that enables the CoNM. Here, one can find the wide range of biologists, neuro-scientists, and doctors caring about state-of-the-art biologic mechanisms, computer scientists and simulation experts caring about the algorithmic processing of simulations, ANN experts caring about the ANN system building, and experts from the specific field of CoNM application (e.g. chemist, physicians, astronomers, engineers). While the NKM model considers this level only for the application of NKM measures, a classical KM model so far has not accessed this kind of role because of the focus on KM measures and activities.

4.5.6 Framework Conditions

The range of conditions of the Potsdam KM Model has been extended by the agenda items of the *strategy and leadership*, the *skills and motivation*, *roles* and *controlling*, which all come from the GPO-WM (Mertens, 2001, p. 109, 108, 107, 113). As these are considered in the neuronal context in addition, the following issues what changes occur in the NKM model and NKMM because of the NKM.

1. *Strategy & Leadership:* In distinction to (1) focusing of measures for an active management supporting an autonomous, self-determined acting of process participants, and (2) acting as coach and the acknowledgment of employees and external knowledge carriers of the GPO-WM (Mertens, 2001, p. 109), the NKM model issues measures for an active management supporting the self-determined acting of ANN systems (3) as well as the acting as coach for ANN-based knowledge carriers (4). Considering all of them, the NKMM considers measures for an

- active management and education by and of ANN systems and human process participants as well.
2. *Organization & Roles:* In addition to measures providing appropriate processes, roles and an internal organization of communities to embed KM services in a company's structure, this is issued by both, the Potsdam KM Model (Gronau, 2009) and the GPO-WM (Mertens, 2001, p. 107), the NKM model addresses measures for the provision of appropriate processes, roles, and communities for the embedding of NKM services. Jointly, the NKMM addresses corresponding measures for the embedding of KM services and NKM services.
 3. *Technique:* As a supplement to the Potsdam KM Model, this limits measures to the establishment of an appropriate infrastructure, software, and hardware, which allows the provision and utilization of KM services, the NKMM considers the provision of an appropriate infrastructure, software and hardware, which allows the provision and utilization of NKM services as well. According to the GPO-WM, this includes an appropriate infrastructure, which supports the communication among process participants, the cooperation and coordination, and the access of information and knowledge (Mertens, 2001, p. 112). The NKM model here only issues the provision of appropriate infrastructure, software, and hardware, which allows the provision and utilization of NKM services only. Resulting infrastructures, software, and hardware differentiates from NKMM models, as the efficient and powerful realization of ML tasks is in focus. Here, knowledge is recognized as object being able to be managed with help of IS (KM- and KL-oriented understanding, cf. section 2.2.1.1.5).
 4. *Culture:* According to the Potsdam KM Model, measures need to address attitudes, qualifications, role understanding and missions of actors of the social system, which all support the utilization of KM services and reduce barriers. In the sense of the NKMM, all of them further need to support the utilization of NKM services. Here, further actors need to be included that have different attitudes, qualifications, role understanding and missions. According to the GPO-WM, measures further need to address the organizational climate, so that a failure tolerant, knowledge exchange friendly, self-determined, and learning friendly environment is supported (Mertens, 2001, p. 111). This recognizes knowledge as processual phenomenon being able to be managed by an adequate environment (KM- and KL-oriented understanding, cf. section 2.2.1.1.5). The NKMM here additionally considers measures supporting a friendly climate for ANN-based knowledge within the whole organization. The NKM model only draws attention to KM services and so appeals a subset of NKMM actors.
 5. *Economic Efficiency & Controlling:* Going beyond measures assuring the beneficial, cost- and time-effective KM service use for the social system actors (1), as

it is demanded by the Potsdam KM Model, and the examination of the successful realization of predefined KM targets (2), as it is demanded by the GPO-WM (Mertens, 2001, p. 113), the NKMM includes the audit of beneficial, cost- and time-effective NKM service use (3) and the fulfillment of predefined NKM targets by hard and soft indicators (4). Here, a harmonized system of objectives is satisfied, while the NKM model draws only attention to (3) and (4).

6. *Rights:* In distinction to measures for the assurance of legal issues, such as property rights of third parties and conditions for governmental approvals about the provision of KM services (1), as well as for the explanation of rights and duties for knowledge workers, that are relevant for the dealing with KM services (2), this is demanded by the Potsdam KM Model, the NKM model issues the compliance of legal issues of NKM services (3) and the explanation of rights and duties for the dealing with NKM services (4). Going beyond, the NKMM draws attention to (1)–(4).
7. *Politics:* In addition to Potsdam KM Model measures providing organizational resources and adjusting legal circumstances, that allow the use of KM services, the NKM model considers the provision of organizational resources and adjustment of legal circumstances, that allow the use of NKM services. Since an integration demands for proper issues, the NKMM draws attention to measures providing organizational resources and adjusting legal circumstances, that allow the use of KM and NKM services.
8. *Skills & Motivation:* Going beyond measures of the GPO-WM (Mertens, 2001, p. 108), that demand for an enhancement of individual skills relevant for business process realizations (1), and the motivation of employees to conduct effective KM activities (2), the NKMM demands for the enhancement of individual skills relevant for neuronal process realizations (3), and the motivation of employees to conduct effective NKM activities (4). The NKM model focuses on (3) and (4) only, so that the ML learning task is privileged and only the dealing with an ANN system motivated.

4.5.7 NKMM Ordering System

In harmony with the Potsdam KM Model (Gronau, 2009, p. 50), the NKMM assumes each KM and NKM measure as well as its activities to satisfy at least one manifestation per dimension. Hence, any KM and NKM measure and its activities can be classified by a cube per phase having the three dimensions of personnel, organizational and procedural range. Fig. 4.13 presents the resulting ordering system, that is represented by two cubes.

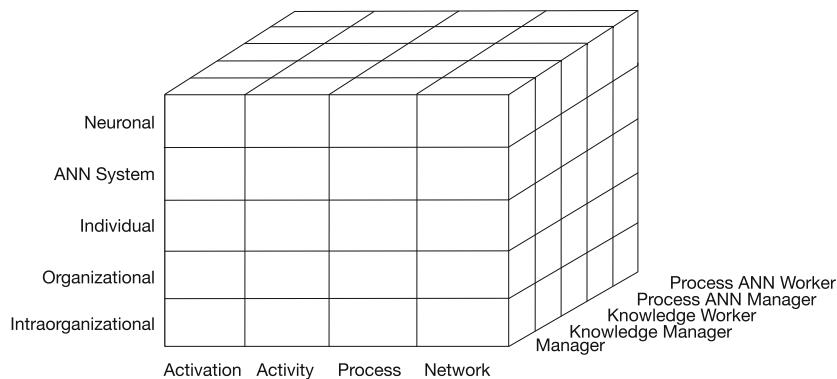


Figure 4.13 The Design of the Neuronal Knowledge Management Ordering System (phase-specific)

A professional KM and NKM guarantees that measures, activities, and tools satisfy all cube components systematically for all phases of a pedagogical valuable, educative learning. The CoNM intends to exemplify the most important components from the perspective of the author, so that the framework functions as a foundation for further research.

4.5.8 Interim Conclusion

The NKM model here designed intends to stand as framework for the management of knowledge constructed by ANN systems. By the provision of concrete NKM activities, a systematic and target-oriented evolution of ANN systems shall be enabled, so that the achievement of a global optimum in learning tasks is supported.

The NKMM here designed intends to stand as framework for the management of knowledge constructed by both organizations and ANN systems. By the provision of KM and NKM activities, a systematic and target-oriented evolution of organizations and ANN systems shall be enabled, so that the achievement of a global optimum in learning organizations is supported.

As these models provide classification blocks according to Fig. 4.13, the models further stand as structuring method for the collection, organization, storing and provision of tools, concepts and methods enabling a neuronal-process-oriented organization. As the systematize the CoNM measures, which result in a neuronal-process-oriented organization, they can be operationalized by the CoNM capable modeling language and guide the functional demonstration of the CoNM.

4.6 Concept of Neuronal Process Modeling

Considering the CoNM meta-model designed in section 4.1, the following technically substantiates modeling objects specified here. By this, the bidirectional exchange of the modeling perspectives and the algorithmic interpretation is enabled, so that both kinds of model constructors are enabled to construct models, humans and machine-based constructors (modeling Def. 5). From hereon, the mixed modeling by different kinds of model constructors in order to create process models through NPM, to create simulation models through NPS and to identify changes within these models, that improve a reference situation through (NPO), is referred to as Concept of Neuronal Process Modeling (CNPM). In distinction to the more general CoNM, it focuses on the modeling of processes by neuronal systems.

Since the dealing with ANN in general is challenged by lacks of interpretability, the CoNM intends to break nontransparent processes down to a set of comprehensible visualizations, so that high dimensional relations are broken down to a two or three dimensional modeling surface without reducing the technical powerfulness of ANN. By considering a vast amount of modeling elements and relations, a CNPM capable modeling language must stand as Process Modeling Language according to Def. 8, Knowledge Modeling Language (Def. 15), Simulation Modeling Language (Def. 18) and ANN Modeling Language (Def. 29), so that the graphical modeling surface established in the following serves as a comprehensible communication base for an algorithmic and experience-based neuronal modeling. It so serves for the following three:

1. for the visualization of algorithmic processing,
2. for the interactive modeling by human constructors,
3. for the visualization of the modeling of machine-based constructors.

Having specified the concrete neuronal model system as a model from an original environment (subjective or objective reality), models constructed on behalf of the CNPM intend to establish a comprehensible visualization on behalf of concrete perspectives (Def. 6) and views (Def. 33). The corresponding relations among original environment, neuronal systems, perspectives, views, and manifestations are visualized in Fig. 4.14. This has been derived from the sub-artifacts as follows.

First, the relation of reality, simulated reality, and models about the reality have been recognized in accordance with Fig. 2.5: As an approximation of the original environment (subjective or objective reality), a huge system of ANN systems intends to represent various of its environmental aspects. It jointly represents an artificial cognitive system. In business contexts, the original environment refers to enterprise

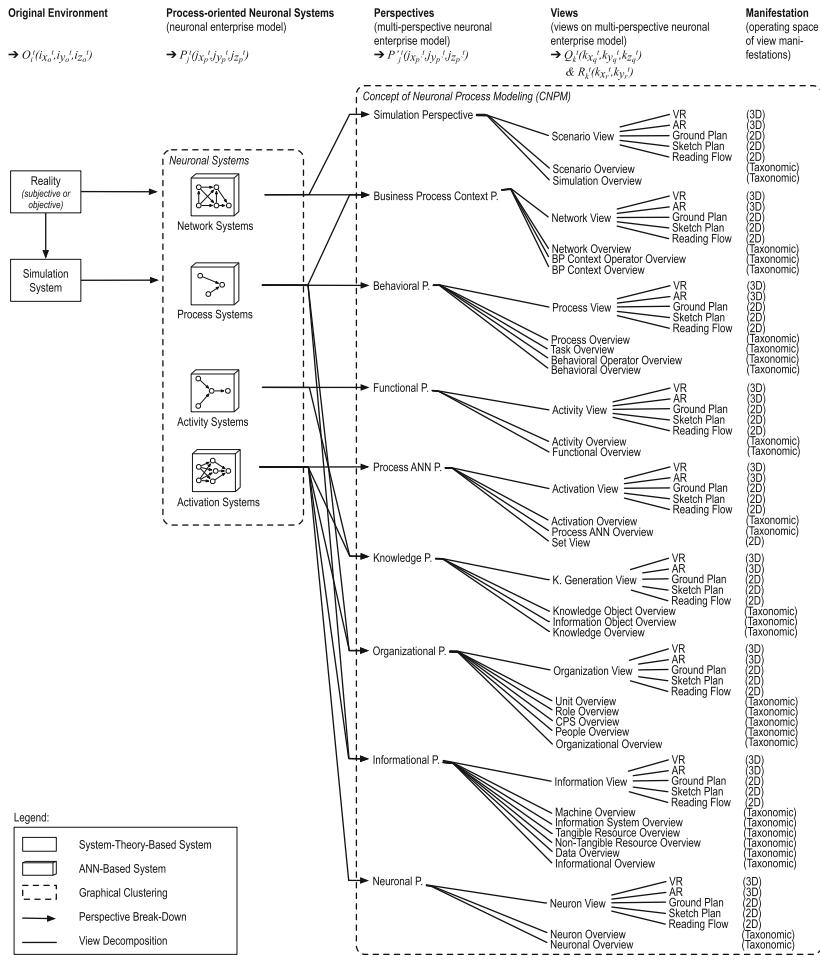


Figure 4.14 Model-ANN-Perspective-View Relations for the CoNM Design (Derived from the CoNM Sub-Artifacts)

complexes. The joint complex of neuronal systems approximates an enterprise incl. its ontological entities e.g. and so forms the neuronal enterprise model (short: NEM, cf. section 4.3). Although the CoNM-capable modeling language, which will be designed by this contribution, focuses on NEM contexts, the CoNM intends to represent non-NEM contexts as well. Here, simply the Context Perspective needs to go beyond BP.

Second, management objects to be approximated by ANNs have been derived from the symbiotic management model designed in section 4.5. Namely, these refer to the activity systems from the procedural range of 1) *process network systems*, 2) *process systems*, 3) *activity systems* and 4) *activation systems*. On behalf of the activation of both, data gathered from the reality and data gathered from the simulation system, the NEM provides the structure and functionality for these kinds of objects being approximated. Hence, these need to be addressed by the neuronal modeling, the neuronal simulation and the neuronal optimization (cf. Fig. 3.1).

Third, perspectives and views have been derived from the CoNM meta-model designed in section 4.1.3. Hence, these follow the perspective definitions presented (Def. 34 – Def. 42). Since the NEM system elements are all assumed to have a spatial characteristic and position, perspectives enable a specialized look on individual aspects of the NEM and they jointly represent the multi-perspective NEM.

Fourth, in accordance with Def. 33 (view), each perspective on the NEM is described by at least one modeling view. Hence, via the look on the NEM as model system, the views intend to jointly clarify circumstances of the reality and form a model system as well. Here, each view considers the viewpoint of one or more stakeholders or ANN instances. The corresponding notation and purpose is to be designed in the following.

Fifth, the CNPM builds on the spatial modeling understanding (Grum and Gronau, 2018a), who establish a knowledge understanding as object-based representation and temporal dynamics of arbitrary complex group of neurons. By this, it is in harmony with the knowledge understanding, that has been established by Def. 13 and the management levels of the organizational range of the NKMM have been addressed (cf. section 4.5).

Sixth, the manifestation has been derived in regard with the operating space of the corresponding view. This can either be in the virtual or real modeling space in two or three dimensions and is in harmony with the bidirectional modeling in any kinds of realities (Grum and Gronau, 2018b). Alternatively, the taxonomic relation of any kinds of activity systems or management levels is addressed (cf. section 4.5).

For the design of the CoNM modeling language, the sub-sections are structured as follows. First, the systems of the NEM are specified. Second, generic statements for the view specification are presented. Then, the views on the NEM are speci-

fied by syntax and modeling convention design. Fourth, the syntax and modeling conventions are extended by modeling shapes and a meta-model for the modeling language for the CoNM is presented. By this, the modeling language of the CoNM is specified. Finally, an interim conclusion draws the association to the initial research questions.

4.6.1 System Specification

In a first step, the different kinds of neuronal systems are specified. Each kind builds on the dynamic model definition of discrete systems of Eq. 2.5, so that temporal effects can be considered when desired.

Since spatial relations are intended to be considered, by a system theoretic modeling, one of the following two ways is possible: First, spatial relations are considered on base of the system theoretic understanding of systems that establish input output relations on base of their neighboring systems. Second, spatial relations are considered by the mathematical modeling according to Eq. 2.6. Although both possibilities are attractive, the following builds on the first option because of the harmonization with discrete systems being considered in simulations. Here, each discrete simulation step can be based on the simulation of an individual system and the system outputs of one time step are considered as system input of the consecutive time step. Only then the high-dimensional operation of an ANN can be objectified for any time step and system. Hence, the following assumes to have for each ANN system a 3D tempo-spatial characterization: Each ANN system $j \in J$ is complemented by an unique spatial and temporal characterization within the 3D Cartesian Coordinate System of Euclidean geometry, such as the by the ordered pair $P_j^t(j_{x_p}^t, j_{y_p}^t, j_{z_p}^t) \mid j_{x_p}^t, j_{y_p}^t, j_{z_p}^t$ are temporal point elements, $j_{x_p}^t, j_{y_p}^t, j_{z_p}^t, t \in \mathbb{R}$ and $j_{x_p}^t, j_{y_p}^t, j_{z_p}^t$ comprise the Cartesian coordinates of the point P at time step t . These represent the original system $i \in I$ having a unique spatial and temporal characterization within the 3D Cartesian Coordinate System, such as the by the ordered pair $O_i^t(i_{x_o}^t, i_{y_o}^t, i_{z_o}^t) \mid i_{x_o}^t, i_{y_o}^t, i_{z_o}^t$ are temporal point elements, $i_{x_o}^t, i_{y_o}^t, i_{z_o}^t, t \in \mathbb{R}$ and $i_{x_o}^t, i_{y_o}^t, i_{z_o}^t$ comprise the Cartesian coordinates of the point O at time step t . Alternative ways for an equivalent characterization refer to the geographic coordinate system, or the Earth-centered Earth-fixed coordinate system, e.g. Since the tempo-spatial information can be transformed among different coordinate systems, the choice of the specific coordinate system is trivial.

Similar to the spatial differentiation, one can handle with substance differentiations in system theory (Imboden and Koch, 2003, p. 81). When dealing with different substances, such as information, material, persons, etc., one principally

can follow one of the following two possibilities. First, one constructs systems of mutually interactive substances. Second, one constructs systems with variables that are coupled by differential equations. As an alternative possibility, one considers all kinds of substances on one common level of interpretation, which is in the CoNM case their interpretation as knowledge. So, for example, each material-based or non-material-based transfer is interpreted as the application of the world's knowledge. Although all possibilities are appealing, the following builds on a mixture of the first and the third option because of Axiom 1. Furthermore, even though each material process instance will be separated by individual systems, each is considered on behalf of the perception and interpretation of the NEM and so, each is integrated with the systems current artificial knowledge base. Only then different kinds of objects can be considered within one ANN system.

4.6.1.1 Network Systems

On the most abstract level, neuronal systems approximate process interdependences. As the interactions of processes are considered, processes are examined in a mesh of networking processes.

Outer effects on the neuronal system designed issue the following:

1. Input from the reality, such as as-is values (unplanned) or to-be-values of current stock of a product, machine data, market price data. Any kind of reality input is modeled as variable $r_i \in Reality$ and we have the set $Reality := \{r_i \mid r \text{ is data input, } i = 1, 2, 3, \dots, n_R, \text{ and } i, n_R, \in \mathbb{N}\}$.
2. Input from the simulation system, such as data coming from a simulated system, the initial configuration of a scenario, or the event-based changes of a scenario within the current system. Any kind of simulation input is modeled as variable $se_j \in SimulationEnvironment$ and we have the set $SimulationEnvironment := \{se_j \mid se \text{ is data input, } j = 1, 2, 3, \dots, n_SE, \text{ and } j, n_SE, \in \mathbb{N}\}$.
3. Input form the super network environment, such as output data required by the super networking process. Any kind of super network environment input is modeled as variable $sne_k \in SuperNetworkEnvironment$ and we have the set $SuperNetworkEnvironment := \{sne_k \mid sne \text{ is data input, } k = 1, 2, 3, \dots, n_SNE, \text{ and } k, n_SNE, \in \mathbb{N}\}$.

Model elements of the neuronal system designed issue the following:

1. Atomic networking processes are considered as ANN systems that approximate the behavior of any kind of object from the reality. These must be considered as model elements. Any kind of networking process is modeled as variable $p_l \in Processes$ and we have the set

$$Processes := \{p_l \mid p \text{ is ANN system}, l = 1, 2, 3, \dots, n_{_P}, \text{ and } l, n_{_P}, \in \mathbb{N}\}.$$
2. Non-atomic networking processes, which are considered as separate ANN systems, referred to as sub-networks, and contain sub-networks of processes. These must be considered as model elements. Any kind of sub-networking process is modeled as variable $sn_m \in SubNetworks$ and we have the set

$$SubNetworks := \{sn_m \mid sn \text{ is ANN system}, m = 1, 2, 3, \dots, n_{_SN}, \text{ and } m, n_{_SN}, \in \mathbb{N}\}.$$
3. Boolean operators, that are considered as separate ANN systems that approximate the behavior of any kind of super network component interplay from the reality in form of a Boolean operator. Hence, it can be interpreted as a special form of a process or sub-scenario. Here, one can find atomic operators, such as an AND split operator having one input and two outputs, an AND join operator having two inputs and one output, an OR split operator having one input and three outputs, etc. These must be considered as model elements of a process context approximating decisions addressing the control flow. Any kind of context operator is modeled as variable $bpcOp_o \in BPCOperators$ and we have the set

$$BPCOperators := \{bpcOp_o \mid bpcOp \text{ is ANN system}, o = 1, 2, 3, \dots, n_{_BPCOP}, \text{ and } o, n_{_BPCOP}, \in \mathbb{N}\}.$$
4. Non-atomic Boolean operators, which are considered as separate ANN systems, referred to as Boolean Sub-Operators, and contain sub-operators and atomic operators. These must be considered as model elements. Any kind of sub-operator is modeled as variable $sBpcOp_o \in SubBPCOperators$ and we have the set

$$SubBPCOperators := \{sBpcOp_p \mid sBpcOp \text{ is ANN system}, p = 1, 2, 3, \dots, n_{_SBpcOP}, \text{ and } p, n_{_SBpcOP}, \in \mathbb{N}\}.$$
5. Control flow edges are considered as inner relation of the model system designed. As weight between ANN systems designed, they connect model elements identified. Any kind of control flow edge is modeled as variable $bpcCf_q \in BPCCControlFlowEdges$ and we have the set

$$BPCCControlFlowEdges := \{bpcCf_q \mid bpcCf \text{ is ANN system connection}, q = 1, 2, 3, \dots, n_{_BPCCF}, \text{ and } q, n_{_BPCCF}, \in \mathbb{N}\}.$$

Considering the entities defined, the network system is specified as model according to Eq. 4.15:

$$\begin{aligned}
 \mathcal{V}_{N_{ANN}}^{(k+1)} &= f(\mathcal{R}_{N_{ANN}}^{(k)}, \mathcal{V}_{N_{ANN}}^{(k)}), \text{ with} \\
 \mathcal{R}_{N_{ANN}}^{(k)} &= \{\text{Reality, Simulation Environment, Super Network Environment}\} \\
 \mathcal{V}_{N_{ANN}}^{(k)} &= \{\text{Processes, BPC Operators, BPC Control Flow Edges,} \\
 &\quad \text{Sub Networks, Sub BPC Operators}\}
 \end{aligned} \tag{4.15}$$

The function $f(\dots)$ refers to the activation of the neuronal system of networking processes, so that the ANN system state is transferred from $\mathcal{V}_{N_{ANN}}^{(k)}$ to $\mathcal{V}_{N_{ANN}}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment of ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design.

Objects are considered as model output if they are used by processes demanded by the reality, if they are used by processes demanded by the simulation environment of the current process or if they are used by processes demanded by the super network environment of the current process.

On behalf of this model definition, a hierarchy of process networks can be established by individual network systems.

4.6.1.2 Process Systems

On the second most abstract level, neuronal systems approximate process instances. As the interactions of processes are not considered, processes are examined individually.

Outer effects on the neuronal system designed issue the following:

1. Input form the network environment, such as output data provided by the super networking process. Any kind of network environment input is modeled as variable $ne_i \in NetworkEnvironment$ and we have the set $NetworkEnvironment := \{ne_i \mid ne \text{ is data input, } i = 1, 2, 3, \dots, n_NE, \text{ and } i, n_NE \in \mathbb{N}\}$.
2. Input form the super process environment, such as output data required by the super process. Any kind of super process environment input is modeled as variable $spe_j \in SuperProcessEnvironment$ and we have the set $SuperProcessEnvironment := \{spe_j \mid spe \text{ is data input, } j = 1, 2, 3, \dots, n_SPE, \text{ and } j, n_SPE \in \mathbb{N}\}$.

Model elements of the neuronal system designed issue the following:

1. Atomic tasks are considered as ANN systems that approximate the behavior of any kind of task from the reality. These must be considered as model elements. Any kind of task is modeled as variable $t_k \in Tasks$ and we have the set $Tasks := \{t_k \mid t \text{ is ANN system, } k = 1, 2, 3, \dots, n_{_T}, \text{ and } k, n_{_T}, \in \mathbb{N}\}$.
2. Non-atomic tasks, which are considered as separate ANN systems, referred to as sub-processes, and contain sub-processes of tasks. These must be considered as model elements. Any kind of sub-process is modeled as variable $sp_l \in SubProcesses$ and we have the set $SubProcesses := \{sp_l \mid sp \text{ is ANN system, } l = 1, 2, 3, \dots, n_{_SP}, \text{ and } l, n_{_SP}, \in \mathbb{N}\}$.
3. Boolean operators, that are considered as separate ANN systems that approximate the behavior of any kind of super task component interplay from the reality in form of a Boolean operator. Hence, it can be interpreted as a special form of a task or sub-process. Here, one can find atomic operators, such as an AND split operator having one input and two outputs, or an AND join operator having two inputs and one output. Further, one can find more complex operators, such as an AND split operator having one input and three outputs — this represents two atomic AND split operators e.g. These must be considered as model elements approximating decisions addressing the control flow. Any kind of operator is modeled as variable $behOp_m \in BehOperators$ and we have the set $BehOperators := \{behOp_m \mid behOp \text{ is ANN system, } m = 1, 2, 3, \dots, n_{_BehOP}, \text{ and } m, n_{_BehOP}, \in \mathbb{N}\}$.
4. Non-atomic Boolean operators, which are considered as separate ANN systems, referred to as sub-Boolean Operators, and contain sub-operators and atomic operators. These must be considered as model elements. Any kind of sub-operator is modeled as variable $sBehOp_o \in SubBehOperators$ and we have the set $SubBehOperators := \{sBehOp_o \mid sBehOp \text{ is ANN system, } o = 1, 2, 3, \dots, n_{_SBehOP}, \text{ and } o, n_{_SBehOP}, \in \mathbb{N}\}$.
5. Control flow edges are considered as inner relation of the model system designed. As weight between ANN systems designed, they connect model elements identified. Any kind of control flow edge is modeled as variable $behCf_p \in BehControlFlowEdges$ and we have the set $BehControlFlowEdges := \{behCf_p \mid behCf \text{ is ANN system connection, } p = 1, 2, 3, \dots, n_{_BehCF}, \text{ and } p, n_{_BehCF}, \in \mathbb{N}\}$.

Considering the entities defined, the network system is specified as model according to Eq. 4.16 shows:

$$\begin{aligned}
 \mathcal{V}_{P_{ANN}}^{(k+1)} &= f(\mathcal{R}_{P_{ANN}}^{(k)}, \mathcal{V}_{P_{ANN}}^{(k)}), \text{ with} \\
 \mathcal{R}_{P_{ANN}}^{(k)} &= \{\text{Network Environment, Super Process Environment}\} \\
 \mathcal{V}_{P_{ANN}}^{(k)} &= \{\text{Tasks, Beh Operators, Beh Control Flow Edges,} \\
 &\quad \text{Sub Processes, Sub Beh Operators}\}
 \end{aligned} \tag{4.16}$$

The function $f(\dots)$ refers to the activation of the neuronal system of individual processes, so that the ANN system state is transferred from $\mathcal{V}_{P_{ANN}}^{(k)}$ to $\mathcal{V}_{P_{ANN}}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment of ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design.

Objects are considered as model output if they are used by processes demanded by the network environment of the current process or if used by processes demanded by the super process environment of the current process.

On behalf of this model definition, a hierarchy of tasks can be established by individual process systems.

4.6.1.3 Activity Systems

On the third most abstract level, neuronal systems approximate activity instances. As the interactions of activities are considered, activities are examined in a mesh of networking activities.

Outer effects on the neuronal system designed issue the following:

1. Input form the process environment, such as output data provided by the super process. Any kind of process environment input is modeled as variable $pe_i \in ProcessEnvironment$ and we have the set $ProcessEnvironment := \{pe_i \mid pe \text{ is data input, } i = 1, 2, 3, \dots, n_{PE}, \text{ and } i, n_{PE}, \in \mathbb{N}\}$.
2. Input form the super activity environment, such as output data required by the super activity. Any kind of super activity environment input is modeled as variable $sae_j \in SuperActivityEnvironment$ and we have the set $SuperActivityEnvironment := \{sae_j \mid sae \text{ is data input, } j = 1, 2, 3, \dots, n_{SAE}, \text{ and } j, n_{SPE}, \in \mathbb{N}\}$.

Model elements of the neuronal system designed issue the following:

1. Atomic activities are considered as ANN systems that approximate the behavior of any kind of activity from the reality. These must be considered as model elements. Any kind of activity is modeled as variable $a_k \in Activities$ and we have the set $Activities := \{a_k \mid a \text{ is ANN system, } k = 1, 2, 3, \dots, n_A, \text{ and } k, n_A, \in \mathbb{N}\}$.

2. Non-atomic activities, which are considered as separate ANN systems, referred to as sub-tasks, and contain sub-tasks of activities. These must be considered as model elements. Any kind of sub-task is modeled as variable $st_l \in SubTasks$ and we have the set $SubTasks := \{st_l \mid st \text{ is ANN system}, l = 1, 2, 3, \dots, n_{ST}, \text{ and } l, n_{ST} \in \mathbb{N}\}$.
3. According to Def. 13, input objects and output objects of ANN systems refer to time-based pattern that objectivize different forms of knowledge, such as knowledge and information. Considered as model elements, any kind of knowledge object is modeled as variable $kom \in KnowledgeObjects$ and we have the set $KnowledgeObjects := \{kom \mid nko \text{ is ANN system pattern}, m = 1, 2, 3, \dots, n_{KO}, \text{ and } m, n_{KO} \in \mathbb{N}\}$. Further, we have for any kind of information object the variable $io_o \in InformationObjects$ and we have the set $InformationObjects := \{io_o \mid io \text{ is ANN system pattern}, o = 1, 2, 3, \dots, n_{IO}, \text{ and } o, n_{IO} \in \mathbb{N}\}$.
4. Knowledge flow edges are considered as inner relation of the model system designed. As weight between ANN systems designed, they connect model elements identified. Any kind of knowledge flow edge is modeled as variable $kfp \in KnowledgeFlowEdges$ and we have the set $KnowledgeFlowEdges := \{kfp \mid kf \text{ is ANN system connection}, p = 1, 2, 3, \dots, n_{KF}, \text{ and } p, n_{KF} \in \mathbb{N}\}$.

Considering the entities defined, the network system is specified as model according to Eq. 4.17 shows:

$$\begin{aligned} \mathcal{V}_{A_{ANN}}^{(k+1)} &= f(\mathcal{R}_{A_{ANN}}^{(k)}, \mathcal{V}_{A_{ANN}}^{(k)}), \text{ with} \\ \mathcal{R}_{A_{ANN}}^{(k)} &= \{ProcessEnvironment, SuperActivityEnvironment\} \\ \mathcal{V}_{A_{ANN}}^{(k)} &= \{Activities, InputObjects, OutputObjects, \\ &\quad KnowledgeFlowEdges, SubTasks\} \end{aligned} \quad (4.17)$$

The function $f(\dots)$ refers to the activation of the neuronal system of individual activities, so that the ANN system state is transferred from $\mathcal{V}_{A_{ANN}}^{(k)}$ to $\mathcal{V}_{A_{ANN}}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment of ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design.

An object is considered as model output if used by activities demanded by the process environment of the current activity or if used by activities demanded by the super activity environment of the current activity.

On behalf of this model definition, a hierarchy of activities can be established by individual activity systems.

4.6.1.4 Activation Systems

On the fourth most abstract level, neuronal systems approximate neuronal activity instances and their neuronal activation. As the interactions of neuronal activities are considered, activities are examined in a mesh of networking neuronal activities.

Outer effects on the neuronal system designed issue the following:

1. Input form the activity environment, such as output data provided by the super activity. Any kind of activity environment input is modeled as variable $ae_i \in ActivityEnvironment$ and we have the set $ActivityEnvironment := \{ae_i \mid ae \text{ is data input, } i = 1, 2, 3, \dots, n_{AE}, \text{ and } i, n_{AE} \in \mathbb{N}\}$.
2. Input form the super neuronal activation environment, such as output data required by the super neuronal activity. Any kind of super neuronal activity environment input is modeled as variable $snae_j \in SuperNeuronalActivity - Environment$ and we have the set $SuperNeuronalActivityEnvironment := \{snae_j \mid snae \text{ is data input, } j = 1, 2, 3, \dots, n_{SNAE}, \text{ and } j, n_{SNAE} \in \mathbb{N}\}$.

Model elements of the neuronal system designed issue the following:

1. Atomic neuronal activities are considered as ANN systems that approximate the behavior of any kind of neuronal activity from the reality. These must be considered as model elements. Any kind of neuronal activity is modeled as variable $na_k \in NeuronalActivities$ and we have the set $NeuronalActivities := \{na_k \mid na \text{ is ANN system, } k = 1, 2, 3, \dots, n_{NA}, \text{ and } k, n_{NA} \in \mathbb{N}\}$.
2. Non-atomic neuronal activities, which are considered as separate ANN systems, referred to as sub-activities, and contain sub-activities of neuronal activities. These must be considered as model elements. Any kind of sub-neuronal activity is modeled as variable $sna_l \in SubActivities$ and we have the set $SubActivities := \{sa_l \mid sa \text{ is ANN system, } l = 1, 2, 3, \dots, n_{SA}, \text{ and } l, n_{SA} \in \mathbb{N}\}$.
3. According to Def. 13, input objects and output objects of ANN systems refer to time-based pattern that objectivize different forms of knowledge, such as neuronal knowledge and neuronal information. Considered as model elements, any kind of neuronal knowledge object is modeled as variable $nko_m \in NeuronalKnowledgeObjects$ and we have the set $NeuronalKnowledgeObjects := \{nko_m \mid nko \text{ is ANN system pattern, } m = 1, 2, 3, \dots, n_{NKO}, \text{ and } m, n_{NKO} \in \mathbb{N}\}$.

$m, n_{NKO}, \in \mathbb{N}\}$. Further, we have for any kind of neuronal information object the variable $nio_o \in NeuronalInformationObjects$ and we have the set $NeuronalInformationObjects := \{nio_o \mid nio \text{ is ANN system pattern, } o = 1, 2, 3, \dots, n_{NIO}, \text{ and } o, n_{NIO}, \in \mathbb{N}\}$.

4. Neuronal knowledge flow edges are considered as inner relation of the model system designed. As the weight between ANN systems designed, they connect model elements identified: they frame neuronal input and output objects and so connect neuronal activities. Any kind of knowledge flow edge is modeled as variable

$nkf_p \in NeuronalKnowledgeFlowEdges$ and we have the set $NeuronalKnowledgeFlowEdges := \{nkf_p \mid nkf \text{ is ANN system connection, } p = 1, 2, 3, \dots, n_{NKF}, \text{ and } p, n_{NKF}, \in \mathbb{N}\}$.

Considering the entities defined, the network system is specified as model according to Eq. 4.18 shows:

$$\begin{aligned}
 \mathcal{V}_{NA_{ANN}}^{(k+1)} &= f(\mathcal{R}_{NA_{ANN}}^{(k)}, \mathcal{V}_{NA_{ANN}}^{(k)}), \text{ with} \\
 \mathcal{R}_{NA_{ANN}}^{(k)} &= \{\text{ActivityEnvironment,} \\
 &\quad \text{SuperNeuronalActivityEnvironment}\} \\
 \mathcal{V}_{NA_{ANN}}^{(k)} &= \{\text{NeuronalActivities, NeuronalInputObjects,} \\
 &\quad \text{NeuronalOutputObjects, NeuronalKnowledgeFlowEdges,} \\
 &\quad \text{SubActivities}\}
 \end{aligned} \tag{4.18}$$

In order to clarify the model elements, the following definitions are presented about neuronal objects:

- *Neuronal knowledge objects* are defined to be neuronal patterns, that evolve as current over a certain period of time that causes a specific behavior of consecutive neurons. Those patterns can reach from single time steps to long periods of time.
- *Neuronal information objects* are defined to be neuronal currents, that serve as interface from and to the environment, such as incoming sensory information and outgoing actuator information. Here, stored information is included as well.
- *Neuronal input objects* are defined to be sensory information objects and knowledge objects.
- *Neuronal output objects* are defined to be actuator information objects and knowledge objects.

Considering those objects, a *neuronal conversion* is defined to be the interplay of different forms of knowledge during the activation of neuronal systems, leading to the transfer of neuronal input objects to neuronal output objects and generates knowledge. Close to the concept of Nonaka and Takeuchi (Nonaka and Takeuchi, 1995), who provide four forms of conversions for the organizational context, the following four forms of neuronal conversion types can be distinguished:

- A *neuronal internalization* is the process of integration of explicit knowledge (neuronal information objects) in tacit knowledge. Here, experiences, and aptitudes are integrated in existing mental models.
- A *neuronal socialization* is the process of experience exchange between neurons within a closed ANN. Here, new tacit knowledge such as common mental models or technical abilities are created.
- A *neuronal externalization* is the process of articulation of tacit knowledge (neuronal knowledge objects) in explicit concepts (neuronal information objects). Here, patterns can serve to verbalize tacit knowledge.
- A *neuronal combination* is the process of connection of available explicit knowledge (neuronal information objects), such that a new explicit knowledge is created. Here, a reorganization, reconfiguration or restructuring can result in new explicit knowledge.

Building on the concept of Gronau and Grum, who provides four levels of a conversion (Gronau, 2012) and neuronal conversion (Grum and Gronau, 2018a), the following four levels of neuronal conversion can be distinguished:

- An *atomic neuronal conversion* is defined to be a neuronal conversion considering only one input object and only one output object.
- *Complex neuronal conversion* are defined to be neuronal conversions considering at least three neuronal objects of one neuron. *Pure* complex neuronal conversions consider only one neuronal conversion type, while *impure* complex neuronal conversions consider several neuronal conversion types, such that one is not able to distinguish them.
- *Abstract neuronal conversion* are defined to be neuronal conversions considering neuronal objects of more than one transferring neuron, such that one is not able to identify atomic knowledge flows of participating neurons because of the abstract level of modeling.
- *Nongranular neuronal conversions* are defined to be neuronal conversions considering neuronal objects of more than one transferring neuron, such that one

cannot identify atomic knowledge flows of participating neurons because of the low granularity level of modeling.

The function $f(\dots)$ refers to the activation of the neuronal system of individual neuronal activities, so that the ANN system state is transferred from $\mathcal{V}_{NA_{ANN}}^{(k)}$ to $\mathcal{V}_{NA_{ANN}}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment of ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design.

An object is considered as model output if used by activities demanded by the activity environment of the current activity or if used by activities of the super neuronal activity environment of the current activity.

On behalf of this model definition, a hierarchy of neuronal activities can be established by individual neuronal activity systems.

4.6.2 Generic View Specification

Before the concrete views are specified in the next subsection, first, generic attributes are presented, that count for any view, second, generic views are presented, that occur in various perspectives of the ANN Process Taxonomy of section 4.1.1. So, redundant information is avoided.

Generic Attributes

The following assumes to have the following attributes enabled for each modeling object represented by a gray rectangular in Fig. 4.4:

- 1) Denomination:** Each model element $k \in K$ of a view is complemented by a unique *denomination* that clarifies the meaning of it. As it is usual in process modeling, the denomination convention of actual entities follows the combination of an object and verb. Further, the denomination convention of the control flow issuing Boolean operators follows the passive mood formed from be + past participle as well as modals + be + past participle.
- 2) 3D tempo-spatial characterization:** Each model element $k \in K$ is complemented by a unique spatial and temporal characterization within the 3D Cartesian coordinate system of Euclidean geometry, such as the by the ordered pair $Q_k^t(k_{x_q}^t, k_{y_q}^t, k_{z_q}^t) \mid k_{x_q}^t, k_{y_q}^t, k_{z_q}^t$ are temporal point elements, $k_{x_q}^t, k_{y_q}^t, k_{z_q}^t, t \in \mathbb{R}$ and $k_{x_q}^t, k_{y_q}^t, k_{z_q}^t$ comprise the Cartesian coordinates of the point Q at time step t . Alternative ways for an equivalent characterization refer to the geographic coordinate system, or the Earth-centered Earth-fixed coordinate system, e.g. In

any case, these follow the 3D tempo-spatial characterization of the corresponding ANN system within the original NEM and $P_j^t(j_{x_p}^t, j_{y_p}^t, j_{z_p}^t) = Q_k^t(k_{x_q}^t, k_{y_q}^t, k_{z_q}^t)$.

3) 3D modeling shape: Further, each model element $k \in K$ can be complemented by a *3D shape* for the augmentation within the real or virtual world. As a 3D shape is provided, the corresponding *2D shapes* can be derived in regard with the viewing angle desired. Since so the 2D ground plan shape, and the 2D sketch plan shape can be derived for example, the visualization within the 2D ground plan, sketch plan and reading flow optimized visualization can be realized.

4) 2D tempo-spatial characterization: For the reading flow optimized visualization, which is from left to right and from top to bottom, the modeling object is complemented by a unique spatial and temporal characterization within the 2D Cartesian coordinate plane, such as the by the ordered pair $R_k^t(k_{x_r}^t, k_{y_r}^t) | k_{x_r}^t, k_{y_r}^t$ are temporal point elements, $k_{x_r}^t, k_{y_r}^t, t \in \mathbb{R}$ and $k_{x_r}^t, k_{y_r}^t$ comprise the Cartesian coordinates of the point R at time step t . As the ground plan and sketch plan are selected for viewing, the corresponding coordinates can be derived from the 3D point $Q_k^t(k_{x_q}^t, k_{y_q}^t, k_{z_q}^t)$. If the reading flow optimized visualization is selected for viewing, the corresponding points either are gathered manually by the human model constructor, or they are gathered by machine-based model constructors, such as ANN modeling agents or algorithms caring about the reading flow and the least possible number of intersecting modeling elements.

Generic Views

The following types of visualizations are stated to be generic since these can be found at different perspectives. So, the following prepares the syntax specification of the next subsection.

1) 3D—VR: On base of the 3D tempo-spatial position of $Q_k^t(k_{x_q}^t, k_{y_q}^t, k_{z_q}^t)$, a 3D view on a display presents a miniature of the NEM (P_j^t) having a variable scale and viewing angle. So, a deeper look on the original environment and O_i^t is enabled.

2) 3D—AR: On base of the 3D tempo-spatial position of $Q_k^t(k_{x_q}^t, k_{y_q}^t, k_{z_q}^t)$, a 3D view on an augmented reality display presents the NEM (P_j^t) having the original scale and viewing angle, so that the modeling items can be displayed next to original environment and O_i^t .

3) 2D—Ground Plan: On base of the 2D tempo-spatial position of $R_k^t(k_{x_r}^t, k_{y_r}^t) = (k_{x_q}^t, k_{y_q}^t)$, a 2D view on a display presents the NEM (P_j^t) having a variable scale and fixed viewing angle, which is from the top. So, a deeper look on the original environment and O_i^t is enabled.

- 4) 2D—Sketch Plan:** On base of the 2D tempo-spatial position of $R_k^t(k_{x_r}^t, k_{y_r}^t) = (k_{x_q}^t, k_{z_q}^t)$, a 2D view on a display presents the NEM (P_j^t) having a variable scale and fixed viewing angle, which is from the front. So, a deeper look on the original environment and O_i^t is enabled.
- 5) 2D—Reading Flow:** On base of the 2D tempo-spatial position of $R_k^t(k_{x_r}^t, k_{y_r}^t)$, a 2D view on a display presents the NEM (P_j^t) being independent from $Q_k^t(k_{x_q}^t, k_{y_q}^t, k_{z_q}^t)$, scale fixed and viewing angle. So, a deeper look on the original environment and O_j^t is enabled.
- 6) Taxonomic Overview:** Disregarding the 3D and 2D temp-spatial positions of any kind of systems, the taxonomic overview associates modeling items by content, so that a hierachic structuring is established per content-related dimension. So, an operating on various levels of details, granularity, and abstraction within the NEM is enabled.

Generic Sets

In the case the sequence of actual entities shall be issued, a flow-oriented kind of view is to be set up that can visualize different forms of flows. In the distinct case the composition of actual entities shall be issued, a taxonomic kind of view is to be set up, that is able to visualize different forms of structure.

Since both kinds of views deal with the visualization of actual entities, they are allowed to consider the same generic kinds of sets and both intend to consider following requirements. First, the visualization shall follow the model definition (Def. 1) considering temporal dynamics; second, the model shall be based on graph theoretic definition (Diestel, 1996); third, the model needs to characterize business processes providing outputs (cf. Fig. 2.3) beside generic processes (cf. Tab. 2.1 for definition). Therefore, the following develops a generalization of a business-process-model-oriented, graph-based description, which can be complemented by a certain context understanding, such as a perspective) and will be used for the specification of any kind of flow-oriented or taxonomic view of the following sections.

As is shown in Fig. 4.15, the different kinds of definitions are brought together on behalf of the establishment of well defined, finite sets.

While the graph theory demands for vertices and edges, the model definition demands for outer effects on a model and its elements, while these do not distinguish between output elements and non-output elements. Hence, the figure visualizes the mathematical modeling, which is developed in the following.

As the model is intended to consider dynamics, the following starts with the time discrete modeling on behalf of Eq. 2.5. Here, the model with the characterization of J , which will show the kind of ANN system, perspective, and view, is defined to be

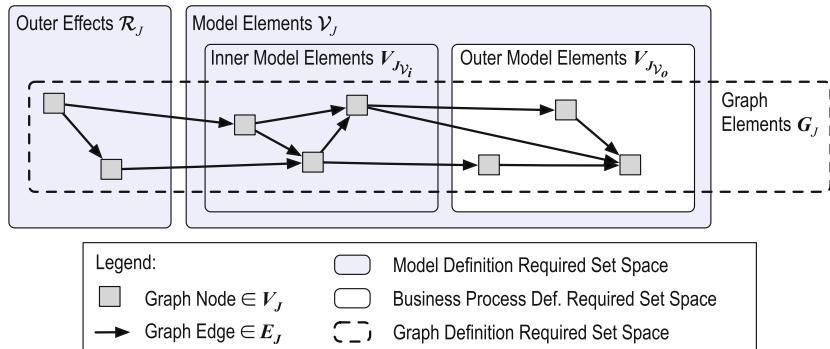


Figure 4.15 The Generic, Model-Oriented, Graph-Based Business Process Description

$$\mathcal{V}_J^{(k+1)} = f(\mathcal{R}_J^{(k)}, \mathcal{V}_J^{(k)}). \quad (4.19)$$

The function $f(\dots)$ issues the transfer from time step k to $k + 1$, \mathcal{R} represents outer effects on the model, and \mathcal{V} stands for the model elements, which includes inner relations as well. Because of the graph theory basement of the model building, the outer effects are assumed to consist of the vertices from $V_{J_{\mathcal{R}}}$ and the edges $E_{J_{\mathcal{R}}}$ among the outer vertices and model elements:

$$\begin{aligned} \mathcal{R}_J^{(k)} &\subseteq V_{J_{\mathcal{R}}} \cup E_{J_{\mathcal{R}}} \\ &\downarrow \\ E_{J_{\mathcal{R}}} &\subseteq (V_{J_{\mathcal{R}}} \times V_{J_{\mathcal{R}}}) \cup (V_{J_{\mathcal{R}}} \times V_{J_{\mathcal{V}}}). \end{aligned}$$

$$\begin{aligned} \mathcal{V}_J^{(k)} &\subseteq V_{J_{\mathcal{V}}} \cup E_{J_{\mathcal{V}}} \\ &\downarrow \\ V_{J_{\mathcal{V}}} &\subseteq V_{J_{V_i}} \cup V_{J_{V_o}} \\ &\downarrow \\ E_{J_{\mathcal{V}}} &\subseteq E_{J_{V_i}} \cup E_{J_{V_o}} \cup (V_{J_{V_i}} \times V_{J_{V_o}}) \\ &\downarrow \\ E_{J_{V_i}} &\subseteq V_{J_{V_i}} \times V_{J_{V_i}} \\ &\downarrow \\ E_{J_{V_o}} &\subseteq V_{J_{V_o}} \times V_{J_{V_o}}. \end{aligned}$$

For a more comprehensible reading, the sets have been indented so that they fit the position of their first occurrence. Model elements $V_J^{(k)}$ are assumed to consist of the vertices from V_{J_V} and the edges E_{J_V} . The latter kind refers to edges among the inner vertices $V_{J_{V_i}}$ (these edges are called $E_{J_{V_i}}$), edges among the outer vertices $V_{J_{V_o}}$ (these edges are called $E_{J_{V_o}}$) and further include edges from inner to outer vertices.

Since any kind of element is built on a vertex-based understanding, vertices are all assumed to represent time-based elements $x^k \in V$, the so called nodes or points, and edges are all assumed to represent directed edges as ordered pairs $e_{1,2} = (x_1^k, x_2^k) \in V^2$, the so called directed edges, directed links, directed lines, arrows or arcs.

As one states the set of all vertices V_J to be $V_J \subseteq V_{J_R} \cup V_{J_V}$ and the set of all edges E_J to be $E_J \subseteq E_{J_R} \cup E_{J_V}$, we can set up the graph-based interpretation G_J for the model characterization called J as follows:

$$G_J = (V_J, E_J). \quad (4.20)$$

Based on this set specification, it is possible to apply different perspectives and construct a modeling language, which is capable to issue different kinds of flows as well as different kinds of structures among networks, processes, activities and activations.

4.6.3 Syntax and Convention Specification

Building on the ANN systems specified in the first subsection, and considering the generic views specified in the subsection before, the concrete views on the NEM are specified according to perspectives defined by the ANN Process Taxonomy of section 4.1.1. By the provision of a modeling syntax and modeling convention, the CoNM-capable modeling language is formalized. So, it becomes clear, which kind of model element is issued per view and how modeling objects are allowed to be connected according to the underlying meta-model (CoNM meta-model designed in section 4.1).

Since the CoNM demands for flow-oriented views as well as taxonomic views, the following sub-sections are oriented to the systematic establishment of views according to corresponding actual entities. As shown in Fig. 4.16, individual sets are covered by the interplay of three different kinds of views systematically.

The *flow-oriented views* address the processual or rather behavioral view on modeling objects of one kind of modeling item set. The *overview* address the view on

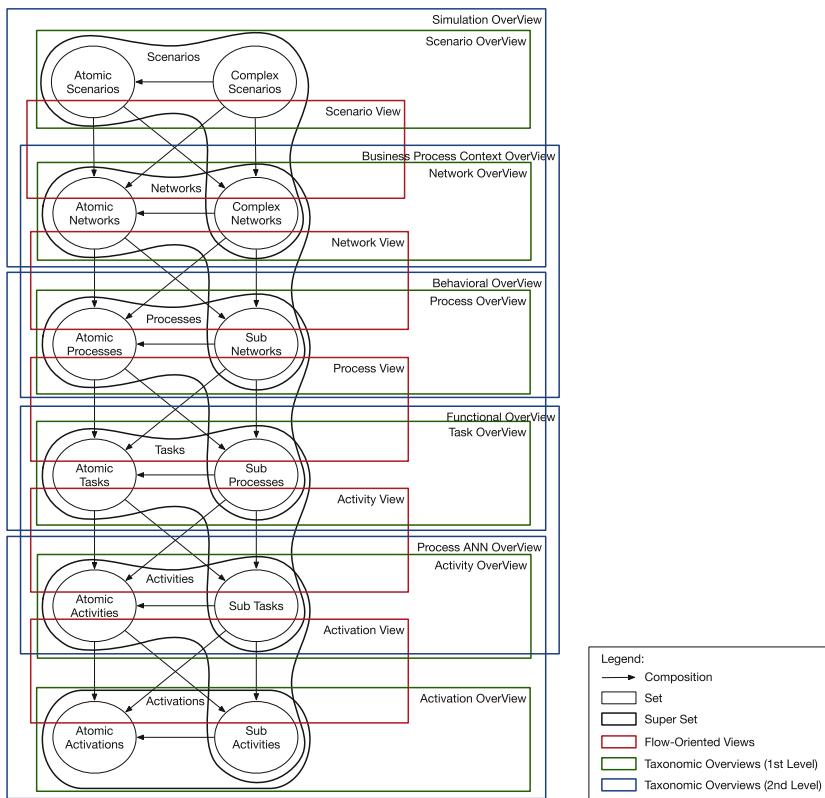


Figure 4.16 The Interrelation of CoNM Views

the hierarchical decomposition of modeling objects. While taxonomic overviews of the first level focus on the view on modeling objects of one kind of modeling item set, the taxonomic overviews of the second level focus on the view on modeling objects of two different kinds of modeling item sets. Hence, the latter kind of taxonomic view enables interlinking of different kinds of modeling item sets.

Any item demanded by the CoNM meta-model, which was not issued as modeling item by one of the views presented in the following sub-sections, is considered as attribute. The attributes are intended to be specified by clicking on the related modeling items and modifying the corresponding table-similar visualization. Later CoNM versions can create views having attributes as modeling items, too.

4.6.3.1 Simulation Views

Based on the CoNM meta-model designed in section 4.1.3, the CoNM modeling language follows the simulation perspective definition presented (Def. 42). In regard with Fig. 4.14, and following the generic view specification of section 4.6.2, the simulation perspective is characterized by the following kinds of views. First, scenario views, and second, simulation overviews. Each of them is specified in the following.

1) Scenario Views

The scenario views on the NEM system are established by the selection of either an atomic or non-atomic network of processes out of the set of network systems (section 4.6.1.1). Since each taxonomy considers scenarios by network process steps from the perspective of the content-related connection of corresponding sequences (network steps) the scenario view of each network selection depicts the precise network processing sequence instance as well as any processing instance alternatives. Due to the strictly formalized hierarchical relation of systems specified, the evolution of processes can be considered from both, the knowledge-management perspective and the ANN perspective.

Since the scenario view is similar to the network view and process view, but having the focus on the concrete instances (jointly considered as the *configuration setting*), the following principally provides the same syntax than the syntax presented in section 4.6.3.2 (network views) and section 4.6.3.3 (process views) and establishes a by-content differentiated view. So, for example all the scenarios are considered that characterize the environmental conditions of a certain process network. Hence, we have

$$\text{ScenarioView} = \{G_{N_{SV_1}}, G_{N_{SV_2}}, G_{N_{SV...}}, G_{N_{SV_n_{SV}}}\} \quad (4.21)$$

with $\text{ScenarioView} \neq \emptyset$.

ScenarioView as Graph: According to the graph-oriented definition of Eq. 4.20, the scenario view establishes the model characterization of N_{SV_i} , which stands for the network system level N taking the scenario view SV and for any process network $i = 1, \dots, n_{SV}$ we set up

$$\begin{aligned} G_{N_{SV}} &= (V_{N_{SV}}, E_{N_{SV}}), \\ &\text{with } V_{N_{SV}} \text{ and } E_{N_{SV}} \end{aligned} \quad (4.22)$$

on base of Eq. 4.29 for network views
and considering scenarios as network
process instances instead of networks.

Please consider here that vertex sets and edge sets therefore follow the same syntax as well as graph vertex and graph edge element definitions as were presented in the corresponding section.

ScenarioView as Model: In order to satisfy the dynamic model definition of discrete systems of Eq. 2.5, and considering the entities defined, the ScenarioView is specified as model $\mathcal{V}_{NSV}^{(k)}$ so that temporal effects can be considered when desired:

$$\begin{aligned}\mathcal{V}_{NSV}^{(k+1)} &= f(\mathcal{R}_{NSV}^{(k)}, \mathcal{V}_{NSV}^{(k)}), \text{ with} \\ \mathcal{R}_{NSV}^{(k)} \text{ and } \mathcal{V}_{NSV}^{(k)} &\text{ on base of Eq. 4.22}\end{aligned}\quad (4.23)$$

following the model-oriented, graph-based

business process description of Eq. 4.19.

The ScenarioView model elements $\mathcal{V}_{NSV}^{(k)}$ therefore consider the scenario graph vertices and edges defined.

Outer effects on the ScenarioView model $\mathcal{R}_{NSV}^{(k)}$ can issue the very same graph elements by principle, too. According to the network system designed in section 4.6.1.1, these must be required by the reality environment or by the super simulation environment as a kind of preceding requirement for the model established, so that $\mathcal{R}_{NSV}^{(k)}$ follows $\mathcal{R}_{NANN}^{(k)}$.

Similar to the determination of a graph vertex or edge to be a model input or an inner model object, the determination of an object to be a model output is carried out: Only if modeling objects are directly considered as model input for consecutively affected scenarios or process networks, they are considered to be a model output of the selected scenario or process network. In the case, scenario outcomes are delivered for a process stakeholder, such as the customer, these are considered as *deliverables*. Following the CoNM meta-model, then, non-tangible resources, such as an oral presentation or knowledge generated, are considered as *service* modeling item. Further, tangible resources, such as a document produced, data generated or components manufactured, are considered as *product*. Hence, these are considered as a kind of enterprise output, department output, or any other kind of ontological level's output.

The function $f(\dots)$ refers to the highlighting of elements within the scenario view model constructed and follows the activation of the neuronal system of individual networks, so that the scenario view state is transferred from $\mathcal{V}_{NSV}^{(k)}$ to $\mathcal{V}_{NSV}^{(k+1)}$ and follows the corresponding transfers of the ANN system state from $\mathcal{V}_{NANN}^{(k)}$ to

$\mathcal{V}_{N_{ANN}}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design. For example, by this, the network of a current simulation thread can be indicated by coloring the corresponding control flow edge.

ScenarioView Conventions: For the ScenarioView of a first version, CoNM-modeling-language-based scenario view models are constructed by the same design rules than the corresponding network views. Please consider scenarios as network instances, here.

ScenarioView Manifestations: The ScenarioView obtains different kinds of manifestations, all of which follow the syntax specified above and consider the generic views specified in section 4.6.2. Hence, we have an ScenarioView in VR, that presents the tempo-spatial positioning of network-induced and process-induced configuration parameters within the virtual reality in 3D. We have an ScenarioView in AR, that presents the tempo-spatial positioning of network-induced and process-induced configuration parameters as augmentation within the real world in 3D. And we have the ScenarioView in 2D, that presents the tempo-spatial positioning of network-induced and process-induced configuration parameters from two defined perspectives (ground plan and sketch plan) as well as its reading optimized visualization.

2) Simulation Overviews

The simulation overviews on the NEM system are established by the selection of an atomic or non-atomic scenario for one atomic or non-atomic network out of the set of network systems (section 4.6.1.1). Since each taxonomy considers scenarios for process networks from the perspective of the content-related connection of scenario sequences (scenario steps), any simulation overview depicts the composition of the configuration settings for processes, networks and its operators with the aid of the taxonomic ordering of corresponding model elements within a certain hierarchy. Because of the strictly formalized hierarchical relation of systems specified, each of them can be specified from both, the knowledge-management perspective and the ANN perspective. Since different forms of hierarchies can be established on base of the same modeling objects, the following speaks rather from a taxonomy than from a hierarchy. Hence, we have

$$\text{SimulationOverView} = \{G_{N_{SOV_1}}, G_{N_{SOV_2}}, G_{N_{SOV_...}}, G_{N_{SOV_{n_{SOV}}}}\}, \quad (4.24)$$

with $\text{SimulationOverView} \neq \emptyset$.

SimulationOverview as Graph: According to the graph-oriented definition of Eq. 4.20, the simulation overview establishes the model characterization of N_{SOV_i} , which stands for the simulation setting of the network system level N taking the simulation overview SOV and we set up for any simulation object $i = 1, \dots, n_{SOV}$

$$\begin{aligned} G_{N_{SOV}} &= (V_{N_{SOV}}, E_{N_{SOV}}), \\ &\text{with} \\ V_{N_{SOV}} &\supseteq \{\text{SimulationTaxonomyVertices}, SB\}, \\ &\text{and} \\ E_{N_{SOV}} &\supseteq \{\text{SimulationTaxonomyEdges}, SBR\}. \end{aligned} \quad (4.25)$$

Please consider here that vertex sets and edge sets are considered as super sets as they represent the theoretically conceivable manifestation of modeling objects. Limitations, as presented by the detailed presentation of its subsets in the following, therefore can be interpreted as syntax of the simulation overview.

The simulation graph vertices $V_{N_{SOV}}$ are further detailed as follows: Since each modeling object j of the vertex sets J can be considered either as input object $j \in V_{N_{SOV_R}}$, as inner object $j \in V_{N_{SOV_{V_i}}}$ or as output object $j \in V_{N_{SOV_{V_o}}}$, for each set specified in the following it holds $J = J_R \cup J_{V_i} \cup J_{V_o}$. Its detailed set construction and syntax can be found at section B.1.

The simulation graph edges $E_{N_{SOV}}$ are further detailed as follows: Since each modeling object j of the edge sets J can be considered either as input object $j \in E_{N_{SOV_R}}$, as inner object $j \in E_{N_{SOV_{V_i}}}$ or as output object $j \in E_{N_{SOV_{V_o}}}$, for each set specified in the following it holds $J = J_R \cup J_{V_i} \cup J_{V_o}$. Its detailed set construction and syntax can be found at section B.2.

SimulationOverview as Model: In order to satisfy the dynamic model definition of discrete systems of Eq. 2.5, and considering the entities defined, the SimulationOverview is specified as model $\mathcal{V}_{N_{SOV}}^{(k)}$ so that temporal effects can be considered when desired:

$$\begin{aligned} \mathcal{V}_{N_{SOV}}^{(k+1)} &= f(\mathcal{R}_{N_{SOV}}^{(k)}, \mathcal{V}_{N_{SOV}}^{(k)}), \text{ with} \\ \mathcal{R}_{N_{SOV}}^{(k)} \text{ and } \mathcal{V}_{N_{SOV}}^{(k)} &\text{ on base of Eq. 4.25} \end{aligned} \quad (4.26)$$

following the model-oriented, graph-based

business process description of Eq. 4.19.

The SimulationOverview model elements $\mathcal{V}_{N_{SOV}}^{(k)}$ therefore consider the simulation graph vertices and edges defined.

Outer effects on the SimulationOverview model $\mathcal{R}_{NSOV}^{(k)}$ can issue the very same graph elements by principle, too. According to the neuronal process system designed in section 4.6.1.1, these either must be required by the reality environment or by the super simulation environment as a kind of preceding requirement for the model established, so that $\mathcal{R}_{NSOV}^{(k)}$ follows $\mathcal{R}_{NANN}^{(k)}$. Please remark here that this goes beyond the isolated view on an individual simulation or scenario constructed by the (theoretically constructed) flow-oriented scenario view, which issues only the question what simulation or scenario input is generated by preceding simulations or scenarios and disregards the question from which simulation or scenario this input is generated. Since the control relation of the scenarios directly follows the control flow specified in the network view (section 4.6.3.2) and process view (section 4.6.3.3), the simulation run can be specified.

Similar to the determination of a graph vertex or edge to be a model input or an inner model object, the determination of an object to be a model output is carried out: Only if modeling objects are directly considered as model input for consecutively affected scenarios (see deliverables of corresponding *ScenarioViews*), the latter kind of scenarios are considered to be a model output of the selected scenario. Hence, any kind of consecutively affected simulation or scenario is considered as part of the model itself and an overview over the current simulation or scenario is set up. Please remark here that this goes beyond the isolated view on an individual simulation or scenario constructed by the flow-oriented scenario view, which issues only the question what simulation or scenario output is generated and disregards the question for which simulation or scenario this output is generated.

The function $f(\dots)$ refers to the highlighting of elements within the simulation overview model constructed and follows the by the reality demanded activation of the neuronal system of individual networks, so that the simulation view state is transferred from $\mathcal{V}_{NSOV}^{(k)}$ to $\mathcal{V}_{NSOV}^{(k+1)}$ and follows the transfer of the ANN system state from $\mathcal{V}_{NANN}^{(k)}$ to $\mathcal{V}_{NANN}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment of ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design. For example, by this, the activation of a current network thread can be indicated by coloring the corresponding simulation vertex, scenario vertex or network vertex.

SimulationOverview Conventions: For the SimulationOverview of a first version, CoNM-modeling-language-based simulation overview models are constructed by the following design rules as they have been summarized in section C.1.

SimulationOverview Manifestations: The simulation overview obtains different kinds of manifestations, all of which follow the syntax specified above. Here, we have the *ScenarioOverView* providing for any simulation scenario object i the graph $G_{N_{ScOV}}$, that issues the entire collection of scenarios:

$$\begin{aligned} \text{ScenarioOverView} &= \{G_{N_{ScOV_1}}, G_{N_{ScOV_2}}, G_{N_{ScOV_...}}, G_{N_{ScOV_{n_{ScOV}}}}\}, \\ &\text{with } G_{N_{ScOV}} = (V_{N_{ScOV}}, E_{N_{ScOV}}) = G_{NSOV}, \\ &\quad \text{having } \text{ScenarioTaxonomyVertices} \neq \emptyset, \\ &\quad \text{and } \text{NetworkTaxonomyVertices} = \emptyset. \end{aligned} \quad (4.27)$$

The corresponding model refers to $\mathcal{V}_{N_{ScOV}}^{(k+1)} = f(\mathcal{R}_{N_{ScOV}}^{(k)}, \mathcal{V}_{N_{ScOV}}^{(k)})$.

In principle, further the *NetworkOverView* can be established. Since this focuses on the networking of the system, which is not the kind perspective intended to be established, here, this view will be defined in Eq. 4.34 of the section addressing the BP context perspective.

Hence, we have the SimulationOverview, that units the taxonomic collection of simulation views and represents more than the individual views.

4.6.3.2 Business Process Context Views

Based on the CoNM meta-model designed in section 4.1.3, the CoNM modeling language follows the Business Process Context (BPC) perspective definition presented (Def. 34). In regard with Fig. 4.14, and following the generic view specification of section 4.6.2, the BPC perspective is characterized by the following kinds of views. First, network views, and second, BPC overviews. Each of them is specified in the following.

1) Network Views

The network views on the NEM system are established by the selection of an atomic or non-atomic network of processes out of the set of network systems (section 4.6.1.1). Since each network considers processes from the perspective of the flow of network sequences (network steps), the network view of each network selection depicts the precise process realization sequence as well as any process realization alternatives. Because of the strictly formalized hierarchical relation of systems specified, tasks can be considered from both, the knowledge-management perspective and the ANN perspective. Hence, we have

$$\text{NetworkView} = \{G_{N_{NV_1}}, G_{N_{NV_2}}, G_{N_{NV_{...}}}, G_{N_{NV_{n_{NV}}}}\} \quad (4.28)$$

with $\text{NetworkView} \neq \emptyset$.

NetworkView as Graph: According to the graph-oriented definition of Eq. 4.20, the network view establishes the model characterization of N_{NV_i} , which stands for the network system level N taking the network view NV and for any network $i = 1, \dots, n_{NV}$ we set up

$$G_{N_{NV}} = (V_{N_{NV}}, E_{N_{NV}}), \quad \text{with}$$

$$V_{N_{NV}} \supseteq \{SB, Processes, SubNetworks, NI, BPC Operators, Types, Events, Roles, Units, PR, IS, Machines, Peoples, CPS, Data, NTR, TR\}, \quad (4.29)$$

and

$$E_{N_{NV}} \supseteq \{CF, MF, NTRF, DF, Z, SBR\}$$

Please consider here that vertex sets and edge sets are considered as super sets as they represent the theoretically conceivable manifestation of modeling objects. Limitations, as presented by the detailed presentation of its subsets in the following, therefore can be interpreted as syntax of the network view.

The network graph vertices $V_{N_{NV}}$ are further detailed as follows: Since each modeling object j of the vertex sets J can be considered either as input object $j \in V_{N_{NV_R}}$, as inner object $j \in V_{N_{NV_{V_i}}}$ or as output object $j \in V_{N_{NV_{V_o}}}$, for each set specified in the following it holds $J = J_R \cup J_{V_i} \cup J_{V_o}$. Its detailed set construction and syntax can be found at section B.3.

The network graph edges $E_{N_{NV}}$ are further detailed as follows: Since each modeling object j of the edge sets J can be considered either as input object $j \in E_{N_{NV_R}}$, as inner object $j \in E_{N_{NV_{V_i}}}$ or as output object $j \in E_{N_{NV_{V_o}}}$, for each set specified in the following it holds $J = J_R \cup J_{V_i} \cup J_{V_o}$. Its detailed set construction and syntax can be found at section B.4.

NetworkView as Model: In order to satisfy the dynamic model definition of discrete systems of Eq. 2.5, and considering the entities defined, the NetworkView is specified as model $\mathcal{V}_{N_{NV}}^{(k)}$ so that temporal effects can be considered when desired:

(4.30)

$$\mathcal{V}_{N_{NV}}^{(k+1)} = f(\mathcal{R}_{N_{NV}}^{(k)}, \mathcal{V}_{N_{NV}}^{(k)}), \text{ with}$$

$$\mathcal{R}_{N_{NV}}^{(k)} \text{ and } \mathcal{V}_{N_{NV}}^{(k)} \text{ on base of Eq. 4.29}$$

following the model-oriented, graph-based
business process description of Eq. 4.19.

The NetworkView model elements $\mathcal{V}_{N_{NV}}^{(k)}$ therefore consider the network graph vertices and edges defined.

Outer effects on the NetworkView model $\mathcal{R}_{N_{NV}}^{(k)}$ can issue the very same graph elements by principle, too. According to the neuronal process system designed in section 4.6.1.1, these either must be required by the simulation environment or by the super network environment, so that $\mathcal{R}_{N_{NV}}^{(k)}$ follows $\mathcal{R}_{N_{ANN}}^{(k)}$.

Similar to the determination of a graph vertex or edge to be a model input or an inner model object, the determination of an object to be a model output is carried out: Only if modeling objects are directly considered as model input for consecutively affected networks, they are considered to be a model output of the selected network. In case network outcomes are delivered for a process stakeholder, such as the customer, these are considered as *deliverables*. Following the CoNM meta-model, then, non-tangible resources, such as an oral presentation or knowledge generated, are considered as *service* modeling item. Further, tangible resources, such as a document produced, data generated or components manufactured, are considered as *product*. Hence, these are considered as a kind of enterprise output, department output, or any other kind of ontological level's output.

The function $f(\dots)$ refers to the highlighting of elements within the network view model constructed and follows the activation of the neuronal system of individual processes, so that the network view state is transferred from $\mathcal{V}_{N_{NV}}^{(k)}$ to $\mathcal{V}_{N_{NV}}^{(k+1)}$ and follows the transfer of the ANN system state from $\mathcal{V}_{N_{ANN}}^{(k)}$ to $\mathcal{V}_{N_{ANN}}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment of ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design. For example, by this, the activation of a current process thread can be indicated by coloring the corresponding control flow edge.

NetworkView Conventions: For the NetworkView of a first version, CoNM-modeling-language-based network models are constructed by the following design rules, which are oriented to the KMDL (section 2.2.3.3). They have been summarized in section C.2.

NetworkView Manifestations: The NetworkView obtains different kinds of manifestations, all of which follow the syntax specified above and consider the generic views specified in section 4.6.2. Hence, we have a NetworkView in VR, that presents the tempo-spatial positioning of networks within the virtual reality in 3D. We have a NetworkView in AR, that presents the tempo-spatial positioning of networks as augmentation within the real world in 3D. And we have the NetworkViews in 2D, that presents the tempo-spatial positioning of networks from two defined perspectives (ground plan and sketch plan) as well as its reading optimized visualization.

2) BP Context Overviews

The BP context overviews on the NEM system are established by the selection of an atomic or non-atomic network out of the set of network systems (section 4.6.1.1). Since each taxonomy considers network steps from the perspective of the content-related connection of network sequences (network steps), any BP context overview depicts the composition of processes, networks and operators with the aid of the taxonomic ordering of corresponding model elements within a certain hierarchy. Because of the strictly formalized hierarchical relation of systems specified, tasks can be considered from both, the knowledge-management perspective and the ANN perspective. Since different forms of hierarchies can be established on base of the same modeling objects, the following speaks rather from a taxonomy than from a hierarchy. Hence, we have

$$\begin{aligned} \text{BPContextOverView} = \\ \{G_{N_{BPCOV_1}}, G_{N_{BPCOV_2}}, G_{N_{BPCOV_{...}}}, G_{N_{BPCOV_{n_{BPCOV}}}}\}, \\ \text{with } \text{BPContextOverView} \neq \emptyset. \end{aligned} \quad (4.31)$$

BPContextOverview as Graph: According to the graph-oriented definition of Eq. 4.20, the BP context overview establishes the model characterization of N_{BPCOV_i} , which stands for the network system level N taking the business process context overview $BPCOV$ and we set up for any context object $i = 1, \dots, n_{BPCOV}$

$$\begin{aligned} G_{N_{BPCOV}} = (V_{N_{BPCOV}}, E_{N_{BPCOV}}), \\ \text{with} \\ V_{N_{BPCOV}} \supseteq \{\text{BPContextTaxonomyVertices}, SB\}, \\ \text{and} \\ E_{N_{BPCOV}} \supseteq \{\text{BPContextTaxonomyEdges}, SBR\}. \end{aligned} \quad (4.32)$$

Please consider here that vertex sets and edge sets are considered as super sets as they represent the theoretically conceivable manifestation of modeling objects. Limitations, as presented by the detailed presentation of its subsets in the following, therefore can be interpreted as syntax of the BP context overview.

The BP context graph vertices $V_{N_{BPCOV}}$ are further detailed as follows: Since each modeling object j of the vertex sets J can be considered either as input object $j \in V_{N_{BPCOV_R}}$, as inner object $j \in V_{N_{BPCOV}_{\mathcal{V}_i}}$ or as output object $j \in V_{N_{BPCOV}_{\mathcal{V}_o}}$, for each set specified in the following it holds $J = J_{\mathcal{R}} \cup J_{\mathcal{V}_i} \cup J_{\mathcal{V}_o}$. Its detailed set construction and syntax can be found at section B.5.

The network graph edges $E_{N_{BPCOV}}$ are further detailed as follows: Since each modeling object j of the edge sets J can be considered either as input object $j \in E_{N_{BPCOV_R}}$, as inner object $j \in E_{N_{BPCOV}_{\mathcal{V}_i}}$ or as output object $j \in E_{N_{BPCOV}_{\mathcal{V}_o}}$, for each set specified in the following it holds $J = J_{\mathcal{R}} \cup J_{\mathcal{V}_i} \cup J_{\mathcal{V}_o}$. Its detailed set construction and syntax can be found at section B.6.

BPContextOverview as Model: In order to satisfy the dynamic model definition of discrete systems of Eq. 2.5, and considering the entities defined, the BPContextOverview is specified as model $\mathcal{V}_{N_{BPCOV}}^{(k)}$ so that temporal effects can be considered when desired:

$$\begin{aligned} \mathcal{V}_{N_{BPCOV}}^{(k+1)} &= f(\mathcal{R}_{N_{BPCOV}}^{(k)}, \mathcal{V}_{N_{BPCOV}}^{(k)}), \text{ with} \\ \mathcal{R}_{N_{BPCOV}}^{(k)} \text{ and } \mathcal{V}_{N_{BPCOV}}^{(k)} &\text{ on base of Eq. 4.32} \\ \text{following the model-oriented, graph-based} \\ \text{business process description of Eq. 4.19.} \end{aligned} \quad (4.33)$$

The BPContextOverview model elements $\mathcal{V}_{N_{BPCOV}}^{(k)}$ therefore consider the BP context graph vertices and edges defined.

Outer effects on the BPContextOverview model $\mathcal{R}_{N_{BPCOV}}^{(k)}$ can issue the very same graph elements by principle, too. According to the neuronal process system designed in section 4.6.1.1, these either must be required by the simulation environment or by the super network environment as a kind of preceding requirement for the model established, so that $\mathcal{R}_{N_{BPCOV}}^{(k)}$ follows $\mathcal{R}_{N_{ANN}}^{(k)}$. Please remark here that this goes beyond the isolated view on an individual networks or processes constructed by the flow-oriented network view, which issues only the question what network or process input is generated by preceding networks or processes and disregards the question from which network or process this input is generated.

Similarly to the determination of a graph vertex or edge to be a model input or an inner model object, the determination of an object to be a model output is

carried out: Only if modeling objects are directly considered as model input for consecutively affected networks (see deliverables of corresponding *NetworkViews*), the latter kind of networks are considered to be a model output of the selected network. Hence, any kind of consecutively affected process or task is considered as part of the model itself and an overview over the current network or process is set up. Please remark here that this goes beyond the isolated view on an individual network or process constructed by the flow-oriented network view, which issues only the question what network or process output is generated and disregards the question for which network or process this output is generated.

The function $f(\dots)$ refers to the highlighting of elements within the BP context overview model constructed and follows the activation of the neuronal system of individual networks, so that the network view state is transferred from $\mathcal{V}_{N_{BPCOV}}^{(k)}$ to $\mathcal{V}_{N_{BPCOV}}^{(k+1)}$ and follows the transfer of the ANN system state from $\mathcal{V}_{N_{ANN}}^{(k)}$ to $\mathcal{V}_{N_{ANN}}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment of ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design. For example, by this, the activation of a current network thread can be indicated by coloring the corresponding network vertex, process vertex or operator vertex.

BPContextOverview Conventions: For the BPContextOverview of a first version, CoNM-modeling-language-based BP context overview models are constructed by the following design rules as they have been summarized in section C.3.

BPContextOverview Manifestations: The BP context overview obtains different kinds of manifestations, all of which follow the syntax specified above. Here, we have the *NetworkOverView* providing for any BP context network object i the graph $G_{N_{NOV}}$, that issues the entire collection of networks:

$$\begin{aligned} \text{NetworkOverView} &= \{G_{N_{NOV_1}}, G_{N_{NOV_2}}, G_{N_{NOV_...}}, G_{N_{NOV_{n_{NOV}}}}\}, \\ \text{with } G_{N_{NOV}} &= (V_{N_{NOV}}, E_{N_{NOV}}) = G_{N_{BPCOV}}, \\ \text{having } \text{NetworkTaxonomyVertices} &\neq \emptyset, \\ \text{and } \text{ProcessTaxonomyVertices} &= \emptyset, \\ \text{and } \text{BPCOperatorTaxonomyVertices} &= \emptyset. \end{aligned} \quad (4.34)$$

The corresponding model refers to $\mathcal{V}_{N_{NOV}}^{(k+1)} = f(\mathcal{R}_{N_{NOV}}^{(k)}, \mathcal{V}_{N_{NOV}}^{(k)})$.

In principle, further the *ProcessOverView* can be established. Since this focuses on the behavior of the system, which is not the kind perspective intended to

be established, here, this view will be defined in Eq. 4.43 of the section addressing the behavioral perspective.

We further have the *BPC Operator OverView* providing for any BP context operator object i the graph $G_{NBP\text{C}oPOV}$, that issues the entire collection of network operators:

$$\begin{aligned} \text{BPC Operator OverView} = \\ \{G_{NBP\text{C}oPOV_1}, G_{NBP\text{C}oPOV_2}, G_{NBP\text{C}oPOV...}, G_{NBP\text{C}oPOV_{nBP\text{C}oPOV}}\}, \\ \text{with } G_{NBP\text{C}oPOV} = (V_{NBP\text{C}oPOV}, E_{NBP\text{C}oPOV}) = G_{NBP\text{COV}}, \\ \text{having } NetworkTaxonomyVertices = \emptyset, \\ \text{and } ProcessTaxonomyVertices = \emptyset, \\ \text{and } BPC\text{ Operator TaxonomyVertices} \neq \emptyset. \end{aligned} \quad (4.35)$$

The corresponding model refers to $\mathcal{V}_{NBP\text{C}oPOV}^{(k+1)} = f(\mathcal{R}_{NBP\text{C}oPOV}^{(k)}, \mathcal{V}_{NBP\text{C}oPOV}^{(k)})$.

Hence, we have the BPContextOverview, that units the taxonomic collection of BP context views and represents more than the individual views.

4.6.3.3 Behavioral Views

Based on the CoNM meta-model designed in section 4.1.3, the CoNM modeling language follows the behavioral perspective definition presented (Def. 35). In regard with Fig. 4.14, and following the generic view specification of section 4.6.2, the behavioral perspective is characterized by the following kinds of views. First, process views, and second, behavioral overviews. Each of them is specified in the following.

1) Process Views

The process views on the NEM system are established by the selection of an atomic or non-atomic process out of the set of process systems (section 4.6.1.2). Since each process considers activities from the perspective of the flow of activity sequences (process steps), the process view of each process selection depicts the precise task processing sequence as well as any task processing alternatives. Because of the strictly formalized hierarchical relation of systems specified, activities can be considered from both, the knowledge-management perspective and the ANN perspective. Hence, we have

$$\begin{aligned} ProcessView = \{G_{PPV_1}, G_{PPV_2}, G_{PPV...}, G_{PPV_{nPPV}}\} \\ \text{with } ProcessView \neq \emptyset. \end{aligned} \quad (4.36)$$

ProcessView as Graph: According to the graph-oriented definition of Eq. 4.20, the process view establishes the model characterization of P_{PV_i} , which stands for the process system level P taking the process view PV and for any process $i = 1, \dots, n_{PV}$ we set up

$$G_{PV} = (V_{PV}, E_{PV}),$$

with

$$V_{PV} \supseteq \{SB, Tasks, SubProcesses, PI, BehOperators, \\ Types, Events, Roles, Units, PR, IS, \\ Machines, Peoples, CPS, Data, NTR, TR\},$$

and

$$E_{PV} \supseteq \{CF, MF, NTRF, DF, Z, SBR\}$$
(4.37)

Please consider here that vertex sets and edge sets are considered as super sets as they represent the theoretically conceivable manifestation of modeling objects. Limitations, as presented by the detailed presentation of its subsets in the following, therefore can be interpreted as syntax of the process view.

The process graph vertices V_{PV} are further detailed as follows: Since each modeling object j of the vertex sets J can be considered either as input object $j \in V_{PV_R}$, as inner object $j \in V_{PV_{V_i}}$ or as output object $j \in V_{PV_{V_o}}$, for each set specified in the following it holds $J = J_R \cup J_{V_i} \cup J_{V_o}$. Its detailed set construction and syntax can be found at section B.7.

The process graph edges E_{PV} are further detailed as follows: Since each modeling object j of the edge sets J can be considered either as input object $j \in E_{PV_R}$, as inner object $j \in E_{PV_{V_i}}$ or as output object $j \in E_{PV_{V_o}}$, for each set specified in the following it holds $J = J_R \cup J_{V_i} \cup J_{V_o}$. Its detailed set construction and syntax can be found at section B.8.

ProcessView as Model: In order to satisfy the dynamic model definition of discrete systems of Eq. 2.5, and considering the entities defined, the ProcessView is specified as model $\mathcal{V}_{PV}^{(k)}$ so that temporal effects can be considered when desired:

$$\mathcal{V}_{PV}^{(k+1)} = f(\mathcal{R}_{PV}^{(k)}, \mathcal{V}_{PV}^{(k)}), \text{ with}$$

$$\mathcal{R}_{PV}^{(k)} \text{ and } \mathcal{V}_{PV}^{(k)} \text{ on base of Eq. 4.37}$$

following the model-oriented, graph-based
business process description of Eq. 4.19.

(4.38)

The ProcessView model elements $\mathcal{V}_{P_{PV}}^{(k)}$ therefore consider the process graph vertices and edges defined.

Outer effects on the ProcessView model $\mathcal{R}_{P_{PV}}^{(k)}$ can issue the very same graph elements by principle, too. According to the neuronal process system designed in section 4.6.1.2, these either must be required by the network environment or by the super process environment, so that $\mathcal{R}_{P_{PV}}^{(k)}$ follows $\mathcal{R}_{P_{ANN}}^{(k)}$.

Similar to the determination of a graph vertex or edge to be a model input or an inner model object, the determination of an object to be a model output is carried out: Only if modeling objects are directly considered as model input for consecutively affected processes, they are considered to be a model output of the selected process. In the case, process outcomes are delivered for a process stakeholder, such as the customer, these are considered as *deliverables*. Following the CoNM meta-model, then, non-tangible resources, such as an oral presentation or knowledge generated, are considered as *service* modeling item. Further, tangible resources, such as a document produced, data generated or components manufactured, are considered as *product*. Hence, these are considered as a kind of enterprise output, department output, or any other kind of ontological level's output.

The function $f(\dots)$ refers to the highlighting of elements within the process view model constructed and follows the activation of the neuronal system of individual processes, so that the process view state is transferred from $\mathcal{V}_{P_{PV}}^{(k)}$ to $\mathcal{V}_{P_{PV}}^{(k+1)}$ and follows the transfer of the ANN system state from $\mathcal{V}_{P_{ANN}}^{(k)}$ to $\mathcal{V}_{P_{ANN}}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment of ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design. For example, by this, the activation of a current process thread can be indicated by coloring the corresponding control flow edge.

ProcessView Conventions: For the ProcessView of a first version, CoNM-modeling-language-based process models are constructed by the following design rules, which are oriented to the KMDL (section 2.2.3.3). They have been summarized in section C.4.

ProcessView Manifestations: The ProcessView obtains different kinds of manifestations, all of which follow the syntax specified above and consider the generic views specified in section 4.6.2. Hence, we have a ProcessView in VR, that presents the tempo-spatial positioning of processes within the virtual reality in 3D. We have a ProcessView in AR, that presents the tempo-spatial positioning of processes as augmentation within the real world in 3D. And we have the ProcessViews in 2D, that presents the tempo-spatial positioning of processes from two defined perspectives (ground plan and sketch plan) as well as its reading optimized visualization.

2) Behavioral Overviews

The behavioral overviews on the NEM system are established by the selection of an atomic or non-atomic process out of the set of process systems (section 4.6.1.2). Since each taxonomy considers process steps from the perspective of the content-related connection of activity sequences (process steps), any behavioral overview depicts the composition of tasks, processes and operators with the aid of the taxonomic ordering of corresponding model elements within a certain hierarchy. Because of the strictly formalized hierarchical relation of systems specified, activities can be considered from both, the knowledge-management perspective and the ANN perspective. Since different forms of hierarchies can be established on base of the same modeling objects, the following speaks rather from a taxonomy than from a hierarchy. Hence, we have

$$\text{BehavioralOverView} = \{G_{P_{BOV_1}}, G_{P_{BOV_2}}, G_{P_{BOV_{..}}}, G_{P_{BOV_{n_{BOV}}}}\}, \quad (4.39)$$

with $\text{BehavioralOverView} \neq \emptyset$.

BehavioralOverview as Graph: According to the graph-oriented definition of Eq. 4.20, the behavioral overview establishes the model characterization of P_{BOV_i} , which stands for the process system level P taking the behavioral overview BOV and we set up for any behavioral object $i = 1, \dots, n_{BOV}$

$$G_{P_{BOV}} = (V_{P_{BOV}}, E_{P_{BOV}}),$$

with

$$V_{P_{BOV}} \supseteq \{\text{BehavioralTaxonomyVertices}, SB\}, \quad (4.40)$$

and

$$E_{P_{BOV}} \supseteq \{\text{BehavioralTaxonomyEdges}, SBR\}.$$

Please consider here that vertex sets and edge sets are considered as super sets as they represent the theoretically conceivable manifestation of modeling objects. Limitations, as presented by the detailed presentation of its subsets in the following, therefore can be interpreted as syntax of the behavioral overview.

The behavioral graph vertices $V_{P_{BOV}}$ are further detailed as follows: Since each modeling object j of the vertex sets J can be considered either as input object $j \in V_{P_{BOV_R}}$, as inner object $j \in V_{P_{BOV_{V_i}}}$ or as output object $j \in V_{P_{BOV_{V_o}}}$, for each set specified in the following it holds $J = J_R \cup J_{V_i} \cup J_{V_o}$. Its detailed set construction and syntax can be found at section B.9.

The process graph edges $E_{P_{BOV}}$ are further detailed as follows: Since each modeling object j of the edge sets J can be considered either as input object $j \in E_{P_{BOV}\mathcal{R}}$, as inner object $j \in E_{P_{BOV}\mathcal{V}_i}$ or as output object $j \in E_{P_{BOV}\mathcal{V}_o}$, for each set specified in the following it holds $J = J_{\mathcal{R}} \cup J_{\mathcal{V}_i} \cup J_{\mathcal{V}_o}$. Its detailed set construction and syntax can be found at section B.10.

BehavioralOverview as Model: In order to satisfy the dynamic model definition of discrete systems of Eq. 2.5, and considering the entities defined, the BehavioralOverview is specified as model $\mathcal{V}_{P_{BOV}}^{(k)}$ so that temporal effects can be considered when desired:

$$\begin{aligned} \mathcal{V}_{P_{BOV}}^{(k+1)} &= f(\mathcal{R}_{P_{BOV}}^{(k)}, \mathcal{V}_{P_{BOV}}^{(k)}), \text{ with} \\ \mathcal{R}_{P_{BOV}}^{(k)} \text{ and } \mathcal{V}_{P_{BOV}}^{(k)} &\text{ on base of Eq. 4.40} \end{aligned} \quad (4.41)$$

following the model-oriented, graph-based
business process description of Eq. 4.19.

The BehavioralOverview model elements $\mathcal{V}_{P_{BOV}}^{(k)}$ therefore consider the behavioral graph vertices and edges defined.

Outer effects on the BehavioralOverview model $\mathcal{R}_{P_{BOV}}^{(k)}$ can issue the very same graph elements by principle, too. According to the neuronal process system designed in section 4.6.1.2, these either must be required by the network environment or by the super process environment as a kind of preceding requirement for the model established, so that $\mathcal{R}_{P_{BOV}}^{(k)}$ follows $\mathcal{R}_{P_{ANN}}^{(k)}$. Please remark here that this goes beyond the isolated view on an individual process or task constructed by the flow-oriented process view, which issues only the question what process or task input is generated by preceding processes or tasks and disregards the question from which process or task this input is generated.

Similar to the determination of a graph vertex or edge to be a model input or an inner model object, the determination of an object to be a model output is carried out: Only if modeling objects are directly considered as model input for consecutively affected processes (see deliverables of corresponding *ProcessViews*), the latter kind of processes are considered to be a model output of the selected process. Hence, any kind of consecutively affected process or task is considered as part of the model itself and an overview over the current processes or task is set up. Please remark here that this goes beyond the isolated view on an individual process or task constructed by the flow-oriented process view, which issues only the question of what process

or task output is generated and disregards the question for which process or task this output is generated.

The function $f(\dots)$ refers to the highlighting of elements within the behavioral overview model constructed and follows the activation of the neuronal system of individual processes, so that the process view state is transferred from $\mathcal{V}_{P_{BOV}}^{(k)}$ to $\mathcal{V}_{P_{BOV}}^{(k+1)}$ and follows the transfer of the ANN system state from $\mathcal{V}_{P_{ANN}}^{(k)}$ to $\mathcal{V}_{P_{ANN}}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment of ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design. For example, by this, the activation of a current process thread can be indicated by coloring the corresponding process vertex, task vertex or operator vertex.

BehavioralOverview Conventions: For the BehavioralOverview of a first version, CoNM-modeling-language-based behavioral overview models are constructed by the following design rules as they have been summarized in section C.5.

BehavioralOverview Manifestations: The Behavioral Overview obtains different kinds of manifestations, all of which follow the syntax specified above. Here, we have the *TaxonomicTaskOverView* providing for any behavioral task object i the graph $G_{P_{TOV}}$, that issues the entire collection of tasks:

$$\begin{aligned} TaskOverView = & \{G_{P_{TOV_1}}, G_{P_{TOV_2}}, G_{P_{TOV_...}}, G_{P_{TOV_{n_{TOV}}}}\}, \\ \text{with } G_{P_{TOV}} = & (V_{P_{TOV}}, E_{P_{TOV}}) = G_{P_{BOV}}, \\ \text{having } TaskTaxonomyVertices & \neq \emptyset, \\ \text{and } ProcessTaxonomyVertices & = \emptyset, \\ \text{and } BehOperatorTaxonomyVertices & = \emptyset. \end{aligned} \quad (4.42)$$

The corresponding model refers to $\mathcal{V}_{P_{TOV}}^{(k+1)} = f(\mathcal{R}_{P_{TOV}}^{(k)}, \mathcal{V}_{P_{TOV}}^{(k)})$.

We further have the *TaxonomicProcessOverView* providing for any behavioral process object i the graph $G_{P_{POV}}$, that issues the entire collection of processes:

$$\begin{aligned} ProcessOverView = & \{G_{P_{POV_1}}, G_{P_{POV_2}}, G_{P_{POV_...}}, G_{P_{POV_{n_{POV}}}}\}, \\ \text{with } G_{P_{POV}} = & (V_{P_{POV}}, E_{P_{POV}}) = G_{P_{BOV}}, \\ \text{having } TaskTaxonomyVertices & = \emptyset, \\ \text{and } ProcessTaxonomyVertices & \neq \emptyset, \\ \text{and } BehOperatorTaxonomyVertices & = \emptyset. \end{aligned} \quad (4.43)$$

The corresponding model refers to $\mathcal{V}_{P_{POV}}^{(k+1)} = f(\mathcal{R}_{P_{POV}}^{(k)}, \mathcal{V}_{P_{POV}}^{(k)})$.

We have the *Taxonomic Behavioral Operator Over View* providing for any behavioral operator object i the graph $G_{P_{BehOpOV}}$, that issues the entire collection of operators:

$$\begin{aligned}
 BehOperatorOverView = & \\
 \{G_{P_{BehOpOV_1}}, G_{P_{BehOpOV_2}}, G_{P_{BehOpOV_{...}}}, G_{P_{BehOpOV_{n_{BehOpOV}}}}\}, \\
 \text{with } G_{P_{BehOpOV}} = (V_{P_{BehOpOV}}, E_{P_{BehOpOV}}) = G_{P_{BOV}}, \\
 & \text{having } TaskTaxonomyVertices = \emptyset, \\
 & \text{and } ProcessTaxonomyVertices = \emptyset, \\
 & \text{and } BehOperatorTaxonomyVertices \neq \emptyset.
 \end{aligned} \tag{4.44}$$

The corresponding model refers to $\mathcal{V}_{P_{BehOpOV}}^{(k+1)} = f(\mathcal{R}_{P_{BehOpOV}}^{(k)}, \mathcal{V}_{P_{BehOpOV}}^{(k)})$.

Hence, we have the BehavioralOverview, that units the taxonomic collection of behavioral views and represents more than the individual views.

4.6.3.4 Functional Views

Based on the CoNM meta-model designed in section 4.1.3, the CoNM modeling language follows the functional perspective definition presented (Def. 36). In regard to Fig. 4.14, and following the generic view specification of section 4.6.2, the functional perspective is characterized by the following kinds of views. First, activity views, and second, functional overviews. Each is specified in the following.

1) Activity Views

The activity views on the NEM system are established by the selection of an atomic or non-atomic activity out of the set of activity systems (section 4.6.1.3). Since each activity considers activations from the perspective of the flow of activation sequences (activity steps), the activity view of each activity selection depicts the precise activation processing sequence as well as any activity processing alternatives. Because of the strictly formalized hierarchical relation of systems specified, activations can be considered from both, the knowledge-management perspective and the ANN perspective. Hence, we have

$$\begin{aligned}
 ActivityView = & \{G_{A_{AV_1}}, G_{A_{AV_2}}, G_{A_{AV_{...}}}, G_{A_{AV_{n_{AV}}}}\} \\
 \text{with } ActivityView \neq \emptyset.
 \end{aligned} \tag{4.45}$$

ActivityView as Graph: According to the graph-oriented definition of Eq. 4.20, the activity view establishes the model characterization of A_{AV_i} , which stands for the activity system level A taking the activity view AV and for any activity $i = 1, \dots, n_{AV}$ we set up

$$G_{AV} = (V_{AV}, E_{AV}),$$

with

$$\begin{aligned} V_{AV} &\supseteq \{SB, Activities, SubTasks, \\ &KB, REQ, Listener, AR, FCT, CM, P, R, Neurons, Types_{Conv}\}, \quad (4.46) \\ &\text{and} \\ E_{AV} &\supseteq \{Z, Conv, SBR\} \end{aligned}$$

Please consider here that vertex sets and edge sets are considered as super sets as they represent the theoretically conceivable manifestation of modeling objects. Limitations, as presented by the detailed presentation of its subsets in the following, therefore can be interpreted as syntax of the activity view.

The activity graph vertices V_{AV} are further detailed as follows: Since each modeling object j of the vertex sets J can be considered either as input object $j \in V_{AV_R}$, as inner object $j \in V_{AV_{V_i}}$ or as output object $j \in V_{AV_{V_o}}$, for each set specified in the following it holds $J = J_R \cup J_{V_i} \cup J_{V_o}$. Its detailed set construction and syntax can be found at section B.11.

The process graph edges E_{AV} are further detailed as follows: Since each modeling object j of the edge sets J can be considered either as input object $j \in E_{AV_R}$, as inner object $j \in E_{AV_{V_i}}$ or as output object $j \in E_{AV_{V_o}}$, for each set specified in the following it holds $J = J_R \cup J_{V_i} \cup J_{V_o}$. Its detailed set construction and syntax can be found at section B.12.

ActivityView as Model: In order to satisfy the dynamic model definition of discrete systems of Eq. 2.5, and considering the entities defined, the ActivityView is specified as model $\mathcal{V}_{AV}^{(k)}$ so that temporal effects can be considered when desired:

$$\begin{aligned} \mathcal{V}_{AV}^{(k+1)} &= f(\mathcal{R}_{AV}^{(k)}, \mathcal{V}_{AV}^{(k)}), \text{ with} \\ \mathcal{R}_{AV}^{(k)} \text{ and } \mathcal{V}_{AV}^{(k)} &\text{ on base of Eq. 4.46} \quad (4.47) \end{aligned}$$

following the model-oriented, graph-based
business process description of Eq. 4.19.

The ActivityView model elements $\mathcal{V}_{AAV}^{(k)}$ therefore consider the activity graph vertices and edges defined.

Outer effects on the ActivityView model $\mathcal{R}_{AAV}^{(k)}$ can issue the very same graph elements by principle, too. According to the neuronal activity system designed in section 4.6.1.3, these either must be required by the process environment or by the super activity environment, so that $\mathcal{R}_{AAV}^{(k)}$ follows $\mathcal{R}_{AANN}^{(k)}$.

Similar to the determination of a graph vertex or edge to be a model input or an inner model object, the determination of an object to be a model output is carried out: Only if modeling objects are directly considered as model input for consecutively affected activities, they are considered to be a model output of the selected activity. In case activity outcomes are delivered for a process stakeholder, such as the customer, these are considered as *deliverables*. Following the CoNM meta-model, then, non-tangible resources, such as an oral presentation or knowledge generated, are considered as *service* modeling item. Further, tangible resources, such as a document produced, data generated or components manufactured, are considered as *product*. Hence, these are considered as a kind of enterprise output, department output, or any other kind of ontological level's output.

The function $f(\dots)$ refers to the highlighting of elements within the activity view model constructed and follows the activation of the neuronal system of individual processes, so that the activity view state is transferred from $\mathcal{V}_{AAV}^{(k)}$ to $\mathcal{V}_{AAV}^{(k+1)}$ and follows the transfer of the ANN system state from $\mathcal{V}_{AANN}^{(k)}$ to $\mathcal{V}_{AANN}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment of ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design. For example, by this, the activation of a current process thread can be indicated by coloring the corresponding control flow edge.

ActivityView Conventions: For the ActivityView of a first version, CoNM-modeling-language-based activity models are constructed by the following design rules, which are oriented to the KMDL (section 2.2.3.3). They have been summarized in section C.6.

ActivityView Manifestations: The ActivityView obtains different kinds of manifestations, all of which follow the syntax specified above and consider the generic views specified in section 4.6.2. Hence, we have an ActivityView in VR, that presents the tempo-spatial positioning of activities within the virtual reality in 3D. We have an ActivityView in AR, that presents the tempo-spatial positioning of activities as augmentation within the real world in 3D. And we have the ActivityViews in 2D, that

presents the tempo-spatial positioning of activities from two defined perspectives (ground plan and sketch plan) as well as its reading optimized visualization.

2) Functional Overviews

The functional overviews on the NEM system are established by the selection of an atomic or non-atomic activities out of the set of activity systems (section 4.6.1.3). Since each taxonomy considers activity steps from the perspective of the content-related connection of activation sequences (activity steps), any functional overview depicts the composition of activities with the aid of the taxonomic ordering of corresponding model elements within a certain hierarchy. Because of the strictly formalized hierarchical relation of systems specified, activations can be considered from both, the knowledge-management perspective and the ANN perspective. Since different forms of hierarchies can be established on base of the same modeling objects, the following speaks rather from a taxonomy than from a hierarchy. Hence, we have

$$\text{FunctionalOverView} = \{G_{AFOV_1}, G_{AFOV_2}, G_{AFOV_{\dots}}, G_{AFOV_{nFOV}}\}, \quad (4.48)$$

with $\text{FunctionalOverView} \neq \emptyset$.

FunctionalOverview as Graph: According to the graph-oriented definition of Eq. 4.20, the functional overview establishes the model characterization of A_{FOV_i} , which stands for the activity system level A taking the functional overview FOV and we set up for any functional object $i = 1, \dots, n_{FOV}$

$$G_{AFOV} = (V_{AFOV}, E_{AFOV}),$$

with

$$V_{AFOV} \supseteq \{\text{FunctionalTaxonomyVertices}, SB\}, \quad (4.49)$$

and

$$E_{AFOV} \supseteq \{\text{FunctionalTaxonomyEdges}, SBR\}.$$

Please consider here that vertex sets and edge sets are considered as super sets as they represent the theoretically conceivable manifestation of modeling objects. Limitations, as they are presented by the detailed presentation of its subsets in the following therefore can be interpreted as syntax of the functional overview.

The functional graph vertices V_{AFOV} are further detailed as follows: Since each modeling object j of the vertex sets J can be considered either as input object $j \in V_{AFOV_R}$, as inner object $j \in V_{AFOV_{V_i}}$ or as output object $j \in V_{AFOV_{V_o}}$, for

each set specified in the following it holds $J = J_{\mathcal{R}} \cup J_{\mathcal{V}_i} \cup J_{\mathcal{V}_o}$. Its detailed set construction and syntax can be found at section B.13.

The activity graph edges E_{AFOV} are further detailed as follows: Since each modeling object j of the edge sets J can be considered either as input object $j \in E_{AFOV_{\mathcal{R}}}$, as inner object $j \in E_{AFOV_{\mathcal{V}_i}}$ or as output object $j \in E_{AFOV_{\mathcal{V}_o}}$, for each set specified in the following it holds $J = J_{\mathcal{R}} \cup J_{\mathcal{V}_i} \cup J_{\mathcal{V}_o}$. Its detailed set construction and syntax can be found at section B.14.

FunctionalOverview as Model: In order to satisfy the dynamic model definition of discrete systems of Eq. 2.5, and considering the entities defined, the FunctionalOverview is specified as model $\mathcal{V}_{AFOV}^{(k)}$ so that temporal effects can be considered when desired:

$$\begin{aligned} \mathcal{V}_{AFOV}^{(k+1)} &= f(\mathcal{R}_{AFOV}^{(k)}, \mathcal{V}_{AFOV}^{(k)}), \text{ with} \\ \mathcal{R}_{AFOV}^{(k)} \text{ and } \mathcal{V}_{AFOV}^{(k)} &\text{ on base of Eq. 4.49} \end{aligned} \quad (4.50)$$

following the model-oriented, graph-based

business process description of Eq. 4.19.

The FunctionalOverview model elements $\mathcal{V}_{AFOV}^{(k)}$ therefore consider the functional graph vertices and edges defined.

Outer effects on the FunctionalOverview model $\mathcal{R}_{AFOV}^{(k)}$ can issue the very same graph elements by principle, too. According to the neuronal process system designed in section 4.6.1.3, these either must be required by the process environment or by the super activity environment as a kind of preceding requirement for the model established, so that $\mathcal{R}_{AFOV}^{(k)}$ follows $\mathcal{R}_{AANN}^{(k)}$. Please remark here that this goes beyond the isolated view on an individual task or activity constructed by the flow-oriented activity view, which issues only the question what task or activity input is generated by preceding tasks or activities and disregards the question from which task or activity this input is generated.

Similar to the determination of a graph vertex or edge to be a model input or an inner model object, the determination of an object to be a model output is carried out: Only if modeling objects are directly considered as model input for consecutively affected tasks (see deliverables of corresponding *ActivityViews*), the latter kind of tasks are considered to be a model output of the selected task. Hence, any kind of consecutively affected task or activity is considered as part of the model itself and an overview over the current task or activity is set up. Please remark here that this goes beyond the isolated view on an individual task or activity constructed by the flow-oriented activity view, which issues only the question of what task or activity

output is generated and disregards the question for which task or activity this output is generated.

The function $f(\dots)$ refers to the highlighting of elements within the functional overview model constructed and follows the activation of the neuronal system of individual activities, so that the activity view state is transferred from $\mathcal{V}_{AOV}^{(k)}$ to $\mathcal{V}_{AOV}^{(k+1)}$ and follows the transfer of the ANN system state from $\mathcal{V}_{ANN}^{(k)}$ to $\mathcal{V}_{ANN}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment of ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design. For example, by this, the activation of a current activity thread can be indicated by coloring the corresponding task vertex or activity vertex.

FunctionalOverview Conventions: For the FunctionalOverview of a first version, CoNM-modeling-language-based functional overview models are constructed by the following design rules as they have been summarized in section C.7.

FunctionalOverview Manifestations: The Functional Overview obtains different kinds of manifestations, all of which follow the syntax specified above. Here, we have the *TaxonomicActivityOverView* providing for any functional activity object i the graph G_{AOV_i} , that issues the entire collection of activities:

$$\begin{aligned} ActivityOverView = & \{G_{AOV_1}, G_{AOV_2}, G_{AOV_{\dots}}, G_{AOV_{n_{AOV}}}\}, \\ & \text{with } G_{AOV} = (V_{AOV}, E_{AOV}) = G_{AOV}, \\ & \text{having } ActivityTaxonomyVertices \neq \emptyset, \\ & \text{and } TaskTaxonomyVertices = \emptyset. \end{aligned} \quad (4.51)$$

The corresponding model refers to $\mathcal{V}_{AOV}^{(k+1)} = f(\mathcal{R}_{AOV}^{(k)}, \mathcal{V}_{AOV}^{(k)})$.

In principle, further the *TaskOverView* can be established. Since this focuses on the behavior of the system, which is not the kind of perspective intended to be established, here, this view will be defined in Eq. 4.42 of the section addressing the behavioral perspective.

Hence, we have the FunctionalOverview, that units the taxonomic collection of functional views and represents more than the individual views.

4.6.3.5 Process ANN Views

Based on the CoNM meta-model designed in section 4.1.3, the CoNM modeling language follows the Process ANN perspective definition presented (Def. 41). In regard to Fig. 4.14, and following the generic view specification of section 4.6.2, the

Process ANN perspective is characterized by the following kinds of views. First, activation views, second, Process ANN overviews, and third, set views. Each of them is specified in the following.

1) Activation Views

The activation views on the NEM system are established by the selection of an atomic or non-atomic activation out of the set of activation systems (section 4.6.1.4). Since each activation considers neuronal activities from the perspective of the flow of activation sequences (activation steps), the activation view of each neuronal activity selection depicts the precise activation processing sequence as well as any activation processing alternatives. Because of the strictly formalized hierarchical relation of systems specified, neuronal activities and activations can be considered on both, the knowledge-management perspective and the ANN perspective. Hence, we have

$$\text{ActivationView} = \{G_{NA_{NAV_1}}, G_{NA_{NAV_2}}, G_{NA_{NAV_{...}}}, G_{NA_{NAV_{n_{NAV}}}}\} \quad (4.52)$$

with $\text{ActivationView} \neq \emptyset$.

ActivationView as Graph: According to the graph-oriented definition of Eq. 4.20, the activation view establishes the model characterization of NA_{NAV_i} , which stands for the neuronal activation system level NA taking the neuronal activation view NAV and for any activation $i = 1, \dots, n_{NAV}$ we set up

$$G_{NA_{NAV}} = (V_{NA_{NAV}}, E_{NA_{NAV}}),$$

with

$$V_{NA_{NAV}} \supseteq \{SB, Activations, SubActivities,$$

$KB, P, R, Neurons, I, D, L, Tr, Te, S, Trainer, NP, DS, Types_{Conv}\},$

and

$$E_{NA_{NAV}} \supseteq \{KF, Z, Conv, SBR\} \quad (4.53)$$

Please consider here that vertex sets and edge sets are considered as super sets as they represent the theoretically conceivable manifestation of modeling objects. Limitations, as presented by the detailed presentation of its subsets in the following, therefore can be interpreted as syntax of the activation view.

The activation graph vertices $V_{NA_{NAV}}$ are further detailed as follows: Since each modeling object j of the vertex sets J can be considered either as input object $j \in V_{NA_{NAV_R}}$, as inner object $j \in V_{NA_{NAV_{Vi}}}$ or as output object $j \in V_{NA_{NAV_{Vo}}}$,

for each set specified in the following it holds $J = J_{\mathcal{R}} \cup J_{\mathcal{V}_i} \cup J_{\mathcal{V}_o}$. Its detailed set construction and syntax can be found at section B.15.

The activation graph edges $E_{NA_{NAV}}$ are further detailed as follows: Since each modeling object j of the edge sets J can be considered either as input object $j \in E_{NA_{NAV_R}}$, as inner object $j \in E_{NA_{NAV}_{\mathcal{V}_i}}$ or as output object $j \in E_{NA_{NAV}_{\mathcal{V}_o}}$, for each set specified in the following it holds $J = J_{\mathcal{R}} \cup J_{\mathcal{V}_i} \cup J_{\mathcal{V}_o}$. Its detailed set construction and syntax can be found at section B.16.

ActivationView as Model: In order to satisfy the dynamic model definition of discrete systems of Eq. 2.5, and considering the entities defined, the ActivationView is specified as model $\mathcal{V}_{NA_{NAV}}^{(k)}$ so that temporal effects can be considered when desired:

$$\begin{aligned} \mathcal{V}_{NA_{NAV}}^{(k+1)} &= f(\mathcal{R}_{NA_{NAV}}^{(k)}, \mathcal{V}_{NA_{NAV}}^{(k)}), \text{ with} \\ \mathcal{R}_{NA_{NAV}}^{(k)} \text{ and } \mathcal{V}_{NA_{NAV}}^{(k)} &\text{ on base of Eq. 4.53} \\ \text{following the model-oriented, graph-based} \\ \text{business process description of Eq. 4.19.} \end{aligned} \quad (4.54)$$

The ActivationView model elements $\mathcal{V}_{NA_{NAV}}^{(k)}$ therefore consider the activation graph vertices and edges defined.

Outer effects on the ActivationView model $\mathcal{R}_{NA_{NAV}}^{(k)}$ can issue the very same graph elements by principle, too. According to the neuronal activation system designed in section 4.6.1.4, these either must be required by the activity environment or by the super neuronal activity environment, so that $\mathcal{R}_{NA_{NAV}}^{(k)}$ follows $\mathcal{R}_{NA_{ANN}}^{(k)}$.

Similar to the determination of a graph vertex or edge to be a model input or an inner model object, the determination of an object to be a model output is carried out: Only if modeling objects are directly considered as model input for consecutively affected activations, they are considered to be a model output of the selected neuronal activity. In case neuronal activity outcomes are delivered for a process stakeholder, such as the customer, these are considered as *deliverables*. Following the CoNM meta-model, then, non-tangible resources, such as an oral presentation or knowledge generated, are considered as *service* modeling item. Further, tangible resources, such as a document produced, data generated or components manufactured, are considered as *product*. Hence, these are considered as a kind of enterprise output, department output, or any other kind of ontological level's output.

The function $f(\dots)$ refers to the highlighting of elements within the activation view model constructed and follows the activation of the neuronal system of individual processes, so that the activation view state is transferred from $\mathcal{V}_{NA_{NAV}}^{(k)}$ to

$\mathcal{V}_{N_{ANAV}}^{(k+1)}$ and follows the transfer of the ANN system state from $\mathcal{V}_{N_{ANN}}^{(k)}$ to $\mathcal{V}_{N_{ANN}}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment of ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design. For example, by this, the activation of a current activity thread can be indicated by coloring the corresponding knowledge flow edge.

ActivationView Conventions: For the ActivationView of a first version, CoNM-modeling-language-based activation models are constructed by the following design rules as they have been summarized in section C.8.

ActivationView Manifestations: The ActivationView obtains different kinds of manifestations, all of which follow the syntax specified above and consider the generic views specified in section 4.6.2. Hence, we have an ActivationView in VR, that presents the tempo-spatial positioning of activations within the virtual reality in 3D. We have an ActivationView in AR that presents the tempo-spatial positioning of activations as augmentation within the real world in 3D. And we have the ActivationViews in 2D that presents the tempo-spatial positioning of activations from two defined perspectives (ground plan and sketch plan) as well as its reading optimized visualization.

2) Process ANN Overviews

The Process ANN overviews on the NEM system are established by the selection of an atomic or non-atomic neuronal activity out of the set of activation systems (section 4.6.1.4). Since each taxonomy considers neuronal activity steps from the perspective of the content-related connection of activation sequences (activation steps), any Process ANN overview depicts the composition of neuronal activities with the aid of the taxonomic ordering of corresponding model elements within a certain hierarchy. Because of the strictly formalized hierarchical relation of systems specified, activations can be considered on both, the knowledge-management perspective and the ANN perspective. Since different forms of hierarchies can be established on base of the same modeling objects, the following speaks rather from a taxonomy than from a hierarchy. Hence, we have

$$\begin{aligned} \text{ProcessANNOverView} = \\ \{G_{NAPANNOV_1}, G_{NAPANNOV_2}, G_{NAPANNOV_{\dots}}, G_{NAPANNOV_{n_{PANNOV}}}\}, \quad (4.55) \\ \text{with } \text{ProcessANNOverView} \neq \emptyset. \end{aligned}$$

ProcessANNOOverview as Graph: According to the graph-oriented definition of Eq. 4.20, the Process ANN overview establishes the model characterization of

NA_{PANNOV_i} , which stands for the neuronal activation system level NA taking the Process ANN overview $PANNOV$ and we set up for any Process ANN object $i = 1, \dots, n_{PANNOV}$

$$G_{NA_{PANNOV}} = (V_{NA_{PANNOV}}, E_{NA_{PANNOV}}), \\ \text{with}$$

$$V_{NA_{PANNOV}} \supseteq \{ProcessANNTaxonomyVertices, SB\}, \\ \text{and}$$

$$E_{NA_{PANNOV}} \supseteq \{ProcessANNTaxonomyEdges, SBR\}.$$

Please consider here that vertex sets and edge sets are considered as super sets as they represent the theoretically conceivable manifestation of modeling objects. Limitations, as presented by the detailed presentation of its subsets in the following, therefore can be interpreted as syntax of the Process ANN overview.

The Process ANN graph vertices $V_{NA_{PANNOV}}$ are further detailed as follows: Since each modeling object j of the vertex sets J can be considered either as input object $j \in V_{NA_{PANNOV_R}}$, as inner object $j \in V_{NA_{PANNOV_V_i}}$ or as output object $j \in V_{NA_{PANNOV_V_o}}$, for each set specified in the following it holds $J = J_R \cup J_{V_i} \cup J_{V_o}$. Its detailed set construction and syntax can be found at section B.17.

The activation graph edges $E_{NA_{PANNOV}}$ are further detailed as follows: Since each modeling object j of the edge sets J can be considered either as input object $j \in E_{NA_{PANNOV_R}}$, as inner object $j \in E_{NA_{PANNOV_V_i}}$ or as output object $j \in E_{NA_{PANNOV_V_o}}$, for each set specified in the following it holds $J = J_R \cup J_{V_i} \cup J_{V_o}$. Its detailed set construction and syntax can be found at section B.18.

ProcessANNOOverview as Model: In order to satisfy the dynamic model definition of discrete systems of Eq. 2.5, and considering the entities defined, the ProcessANNOOverview is specified as model $\mathcal{V}_{NA_{PANNOV}}^{(k)}$ so that temporal effects can be considered when desired:

$$\begin{aligned} \mathcal{V}_{NA_{PANNOV}}^{(k+1)} &= f(\mathcal{R}_{NA_{PANNOV}}^{(k)}, \mathcal{V}_{NA_{PANNOV}}^{(k)}), \text{ with} \\ \mathcal{R}_{NA_{PANNOV}}^{(k)} \text{ and } \mathcal{V}_{NA_{PANNOV}}^{(k)} &\text{ on base of Eq. 4.56} \\ \text{following the model-oriented, graph-based} \\ \text{business process description of Eq. 4.19.} \end{aligned} \quad (4.57)$$

The ProcessANNOOverview model elements $\mathcal{V}_{NAPANNOV}^{(k)}$ therefore consider the Process ANN graph vertices and edges defined.

Outer effects on the ProcessANNOOverview model $\mathcal{R}_{NAPANNOV}^{(k)}$ can issue the very same graph elements by principle, too. According to the neuronal activation system designed in section 4.6.1.4, these either must be required by the activity environment or by the super activation environment as a kind of preceding requirement for the model established, so that $\mathcal{R}_{NAPANNOV}^{(k)}$ follows $\mathcal{R}_{NA_{ANN}}^{(k)}$. Please remark here that this goes beyond the isolated view on an individual activity or activation constructed by the flow-oriented activation view, which issues only the question what activity or activation input is generated by preceding activities or activations and disregards the question from which activity or activation this input is generated.

Similar to the determination of a graph vertex or edge to be a model input or an inner model object, the determination of an object to be a model output is carried out: Only if modeling objects are directly considered as model input for consecutively affected activities (see deliverables of corresponding *ActivationViews*), the latter kind of activities are considered to be a model output of the selected activities. Hence, any kind of consecutively affected activity or activation is considered as part of the model itself and an overview over the current activity or activation is set up. Please remark here that this goes beyond the isolated view on an individual activity or activation constructed by the flow-oriented activation view, which issues only the question what activity or activation output is generated and disregards the question for which activity or activation this output is generated.

The function $f(\dots)$ refers to the highlighting of elements within the Process ANN overview model constructed and follows the activation of the neuronal system of individual activations, so that the activation view state is transferred from $\mathcal{V}_{NAPANNOV}^{(k)}$ to $\mathcal{V}_{NAPANNOV}^{(k+1)}$ and follows the transfer of the ANN system state from $\mathcal{V}_{NA_{ANN}}^{(k)}$ to $\mathcal{V}_{NA_{ANN}}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment of ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design. For example, by this, the activation of a current activation thread can be indicated by coloring the corresponding activity vertex or activation vertex.

ProcessANNOOverview Conventions: For the ProcessANNOOverview of a first version, CoNM-modeling-language-based Process ANN overview models are constructed by the following design rules as they have been summarized in section C.9.

ProcessANNOOverview Manifestations: The Process ANN Overview obtains different kinds of manifestations, all of which follow the syntax specified above. Here,

we have the *Taxonomic Activation Over View* providing for any neuronal activity object i the graph $G_{NA_{NAOV}}$, that issues the entire collection of Activations:

$$\begin{aligned} ActivationOverView = & \{G_{NA_{NAOV_1}}, G_{NA_{NAOV_2}}, G_{NA_{NAOV_...}}, G_{NA_{NAOV_{n_{NAOV}}}}\}, \\ & \text{with } G_{NA_{NAOV}} = (V_{NA_{NAOV}}, E_{NA_{NAOV}}) = G_{NA_{PANNOV}}, \\ & \quad \text{having } ActivationTaxonomyVertices \neq \emptyset, \\ & \quad \text{and } ActivityTaxonomyVertices = \emptyset. \end{aligned} \quad (4.58)$$

The corresponding model refers to $\mathcal{V}_{NA_{NAOV}}^{(k+1)} = f(\mathcal{R}_{NA_{NAOV}}^{(k)}, \mathcal{V}_{NA_{NAOV}}^{(k)})$.

In principle, further the *ActivityOverView* can be established. Since this focuses on the function of the system, which is not the kind perspective intended to be established, here, this view is defined in Eq. 4.51 of the section addressing the functional perspective.

Hence, we have the *ProcessANNOOverview*, that units the taxonomic collection of Process ANN views and represents more than the individual views.

3) Set Views

The set views on the NEM system are established by the selection at least one and at maximum two atomic or non-atomic activities out of the set of activity systems (section 4.6.1.3) or of at least one and at maximum two atomic or non-atomic activations out of the set of activation systems (section 4.6.1.4). Since each set considers activities or rather neuronal activities from the perspective of the flow of corresponding sequences (activity or activation steps), the set view of each activity or activation selection depicts the precise knowledge compilation of input and output objects for any activity or activation processing sequence as well as any corresponding processing alternatives. Because of the strictly formalized hierarchical relation of systems specified, sets for neuronal activities and activations can be considered on both the knowledge-management perspective and the ANN perspective. Hence, we have

$$SetView = \{G_{AandNA_{SV_1}}, G_{AandNA_{SV_2}}, G_{AandNA_{SV_...}}, G_{AandNA_{SV_{n_{SV}}}}\} \quad (4.59)$$

with $SetView \neq \emptyset$.

SetView as Graph: According to the graph-oriented definition of Eq. 4.20, the set view establishes the model characterization of $AandNA_{SV_i}$, which stands for the activity and neuronal activation system level $AandNA$ taking the set view SV and for any data set $i = 1, \dots, n_{SV}$ we set up

$$G_{AandNASV} = (V_{AandNASV}, E_{AandNASV}),$$

with

$$V_{AandNASV} \supseteq \{SB, KB, P, R, Neurons\}, \quad (4.60)$$

and

$$E_{AandNASV} \supseteq \{Z, SBR\}$$

Please consider here that vertex sets and edge sets are considered as super sets as they represent the theoretically conceivable manifestation of modeling objects. Limitations, as presented by the detailed presentation of its subsets in the following, therefore can be interpreted as syntax of the set view.

The set graph vertices $V_{AandNASV}$ are further detailed as follows: Since each modeling object j of the vertex sets J can be considered either as input object $j \in V_{AandNASV_R}$, as inner object $j \in V_{AandNASV_{V_i}}$ or as output object $j \in V_{AandNASV_{V_o}}$, for each set specified in the following it holds $J = J_R \cup J_{V_i} \cup J_{V_o}$. Its detailed set construction and syntax can be found at section B.19.

The set graph edges $E_{AandNASV}$ are further detailed as follows: Since each modeling object j of the edge sets J can be considered either as input object $j \in E_{AandNASV_R}$, as inner object $j \in E_{AandNASV_{V_i}}$ or as output object $j \in E_{AandNASV_{V_o}}$, for each set specified in the following it holds $J = J_R \cup J_{V_i} \cup J_{V_o}$. Its detailed set construction and syntax can be found at section B.20.

SetView as Model: In order to satisfy the dynamic model definition of discrete systems of Eq. 2.5, and considering the entities defined, the SetView is specified as model $\mathcal{V}_{AandNASV}^{(k)}$ so that temporal effects can be considered when desired:

$$\begin{aligned} \mathcal{V}_{AandNASV}^{(k+1)} &= f(\mathcal{R}_{AandNASV}^{(k)}, \mathcal{V}_{AandNASV}^{(k)}), \text{ with} \\ \mathcal{R}_{AandNASV}^{(k)} \text{ and } \mathcal{V}_{AandNASV}^{(k)} &\text{ on base of Eq. 4.60} \\ \text{following the model-oriented, graph-based} \\ \text{business process description of Eq. 4.19.} \end{aligned} \quad (4.61)$$

The SetView model elements $\mathcal{V}_{AandNASV}^{(k)}$ therefore consider the set graph vertices and edges defined.

Outer effects on the SetView model $\mathcal{R}_{AandNASV}^{(k)}$ can issue the very same graph elements by principle, too. According to the activity and activation system designed in section 4.6.1.3 and section 4.6.1.4, these either must be required by the process environment or by the super activity environment, so that $\mathcal{R}_{AandNASV}^{(k)}$ follows

$\mathcal{R}_{AANN}^{(k)}$ or these either must be required by the activity environment or by the super neuronal activity environment, so that $\mathcal{R}_{AandNASV}^{(k)}$ follows $\mathcal{R}_{NAANN}^{(k)}$.

Similar to the determination of a graph vertex or edge to be a model input or an inner model object, the determination of an object to be a model output is carried out: Only if modeling objects are directly considered as model input for consecutively affected sets, they are considered to be a model output of the selected set. In case set outcomes are delivered for a process stakeholder, such as the customer, these are considered as *deliverables*. Following the CoNM meta-model, then, non-tangible resources, such as an oral presentation or knowledge generated, are considered as *service* modeling item. Further, tangible resources, such as a document produced, data generated or components manufactured, are considered as *product*. Hence, these are considered as a kind of enterprise output, department output, or any other kind of ontological level's output.

The function $f(\dots)$ refers to the highlighting of elements within the set view model constructed and follows the activation of the neuronal system of individual processes and activities, so that the set view state is transferred from $\mathcal{V}_{AandNASV}^{(k)}$ to $\mathcal{V}_{AandNASV}^{(k+1)}$ and follows the transfer of the ANN system state from $\mathcal{V}_{AANN}^{(k)}$ to $\mathcal{V}_{AANN}^{(k+1)}$ or rather it follows the transfer of the ANN system state from $\mathcal{V}_{NAANN}^{(k)}$ to $\mathcal{V}_{NAANN}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment of ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design. For example, by this, the activation of a current activity thread or activation set can be indicated by coloring the corresponding (1) knowledge object, (2) information object, (3) mixture object, (4) moment, or (5) sequence.

SetView Conventions: For the SetView of a first version, CoNM-modeling-language-based knowledge generation view models are constructed by the same design rules than the corresponding activity or rather activation views.

SetView Manifestations: The SetView obtains different kinds of manifestations, all of which follow the syntax specified above and consider the generic views specified in section 4.6.2. Hence, we have the SetView in 2D that presents the following two kinds of view manifestations:

The first kind of 2D manifestations presents the reading optimized visualization of a set by the visualization of the set view's items similar to a matrix notation. Here, modeling objects either list set elements vertically or horizontally as a transposed version, so that the set composition of the moment's elements and the moment composition to sequences becomes clear.

The second kind of 2D manifestations presents the tempo-spatial positioning of sets from two defined perspectives (ground plan and sketch plan), so that a set composition is prepared for a 3D visualization. Here, e.g. baseline items and target items are arranged by a x-axis, so that the corresponding moment is characterized. Moments might be arranged by a y-axis, so that the corresponding sequence is characterized. Finally, sequences can be arranged by a z-axis, so that the corresponding data set is characterized.

Further, we have a SetView in VR that presents the tempo-spatial positioning of sets within the virtual reality in 3D. This follows the x-, y- and z-axis arrangements prepared by the ground and sketch plan.

Finally, we have a SetView in AR that presents the tempo-spatial positioning of sets as augmentation within the real world in 3D. This follows the x-, y-, and z-axis arrangements prepared by the ground and sketch plan and positions the joint data set next to its activation defined in its corresponding ActivationView.

4.6.3.6 Knowledge Views

Based on the CoNM meta-model designed in section 4.1.3, the CoNM modeling language follows the knowledge perspective definition presented (Def. 38). In regard with Fig. 4.14, and following the generic view specification of section 4.6.2, the Knowledge perspective is characterized by the following kinds of views. First, knowledge generation views, and second, knowledge overviews. Each of them is specified in the following.

1) Knowledge Generation Views

The knowledge generation views on the NEM system are established by the selection of either an atomic or non-atomic activity out of the set of activity systems (section 4.6.1.3) or by the selection of an atomic or non-atomic activation out of the set of activation systems (section 4.6.1.4). Since each taxonomy considers knowledge generated by activity steps or rather activation steps from the perspective of the content-related connection of corresponding sequences (activity or activation steps), the knowledge generation view of each knowledge selection depicts the precise activity-induced or activation-induced knowledge processing sequence as well as any processing alternatives. Due to the strictly formalized hierarchical relation of systems specified, the generation can be considered from both, the knowledge-management perspective and the ANN perspective.

Since the knowledge generation view is similar to activity view and activation view, but having the focus on knowledge distribution, this includes its generation and use in activities or rather activations, the following principally provides the same syntax than the syntax presented in section 4.6.3.4 (activity views) and sec-

tion 4.6.3.5 (activation views) and establishes a by-content differentiated view. So, for example all the activities and activations are considered that distribute and reuse a certain knowledge object or information object. Hence, we have

$$\begin{aligned} \text{KnowledgeGenerationView} = \\ \{G_{AandNA_{KGV_1}}, G_{AandNA_{KGV_2}}, G_{AandNA_{KGV...}}, G_{AandNA_{KGV_{n_{KGV}}}}\} \quad (4.62) \\ \text{with } \text{KnowledgeGenerationView} \neq \emptyset. \end{aligned}$$

KnowledgeGenerationView as Graph: According to the graph-oriented definition of Eq. 4.20, the knowledge generation view establishes the model characterization of $AandNA_{KGV_i}$, which stands for the activity system level A and the activation system level NA taking the knowledge generation view KGV and for any activity-induced or rather activation-induced knowledge object $i = 1, \dots, n_{KGV}$ we set up

$$\begin{aligned} G_{AandNA_{KGV}} = (V_{AandNA_{KGV}}, E_{AandNA_{KGV}}), \\ \text{with} \\ V_{AandNA_{KGV}} \text{ and } E_{AandNA_{KGV}} \quad (4.63) \\ \text{on base of Eq. 4.46 for activity views} \\ \text{and on base of Eq. 4.53 for activation views.} \end{aligned}$$

Please consider here that vertex sets and edge sets therefore follow the same syntax as well as graph vertex and graph edge element definitions as were presented in the corresponding sections.

KnowledgeGenerationView as Model: In order to satisfy the dynamic model definition of discrete systems of Eq. 2.5, and considering the entities defined, the KnowledgeGenerationView is specified as model $\mathcal{V}_{AandNA_{KGV}}^{(k)}$ so that temporal effects can be considered when desired:

$$\begin{aligned} \mathcal{V}_{AandNA_{KGV}}^{(k+1)} = f(\mathcal{R}_{AandNA_{KGV}}^{(k)}, \mathcal{V}_{AandNA_{KGV}}^{(k)}), \text{ with} \\ \mathcal{R}_{AandNA_{KGV}}^{(k)} \text{ and } \mathcal{V}_{AandNA_{KGV}}^{(k)} \text{ on base of Eq. 4.63} \\ \text{following the model-oriented, graph-based} \\ \text{business process description of Eq. 4.19.} \quad (4.64) \end{aligned}$$

The KnowledgeGenerationView model elements $\mathcal{V}_{AandNA_{KGV}}^{(k)}$ therefore consider the knowledge generation graph vertices and edges defined.

Outer effects on the KnowledgeGenerationView model $\mathcal{R}_{AandNA_{KGV}}^{(k)}$ can issue the very same graph elements by principle, too. According to the activity system designed in section 4.6.1.3, and according to the neuronal activation system designed in section 4.6.1.4, these either must be required by the task environment or by the super activity environment as a kind of preceding requirement for the model established, so that $\mathcal{R}_{AandNA_{KGV}}^{(k)}$ follows $\mathcal{R}_{A_{ANN}}^{(k)}$, or alternatively, these must be required by the activity environment or by the super activation environment as a kind of preceding requirement for the model established, so that $\mathcal{R}_{AandNA_{KGV}}^{(k)}$ follows $\mathcal{R}_{NA_{ANN}}^{(k)}$.

Similar to the determination of a graph vertex or edge to be a model input or an inner model object, the determination of an object to be a model output is carried out: Only if modeling objects are directly considered as model input for consecutively affected activities or activations, they are considered to be a model output of the selected activity or activation. In case knowledge outcomes are delivered for a process stakeholder, such as the customer, these are considered as *deliverables*. Following the CoNM meta-model, then, non-tangible resources, such as an oral presentation or knowledge generated, are considered as *service* modeling item. Further, tangible resources, such as a document produced, data generated or components manufactured, are considered as *product*. Hence, these are considered as a kind of enterprise output, department output, or any other kind of ontological level's output.

The function $f(\dots)$ refers to the highlighting of elements within the knowledge generation view model constructed and follows the activation of the neuronal system of individual activities and activations, so that the knowledge view state is transferred from $\mathcal{V}_{AandNA_{KGV}}^{(k)}$ to $\mathcal{V}_{AandNA_{KGV}}^{(k+1)}$ and follows the corresponding transfers of the ANN system state from $\mathcal{V}_{A_{ANN}}^{(k)}$ to $\mathcal{V}_{A_{ANN}}^{(k+1)}$ and from $\mathcal{V}_{NA_{ANN}}^{(k)}$ to $\mathcal{V}_{NA_{ANN}}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design. For example, by this, the activation of a current process thread can be indicated by coloring the corresponding control flow edge.

KnowledgeGenerationView Conventions: For the KnowledgeGenerationView of a first version, CoNM-modeling-language-based knowledge generation view models are constructed by the same design rules than the corresponding activity or rather activation views.

KnowledgeGenerationView Manifestations: The KnowledgeGenerationView obtains different kinds of manifestations, all of which follow the syntax specified

above and consider the generic views specified in section 4.6.2. Hence, we have an KnowledgeGenerationView in VR, that presents the tempo-spatial positioning of activity-induced and activation-induced knowledge within the virtual reality in 3D. We have an KnowledgeGenerationView in AR that presents the tempo-spatial positioning of activity-induced and activation-induced knowledge as augmentation within the real world in 3D. And we have the KnowledgeGenerationViews in 2D that presents the tempo-spatial positioning of activity-induced and activation-induced knowledge from two defined perspectives (ground plan and sketch plan) as well as its reading optimized visualization.

2) Knowledge Overviews

The knowledge overviews on the NEM system are established by the selection of either an atomic or non-atomic activity out of the set of activity systems (section 4.6.1.3) or an atomic or non-atomic neuronal activity out of the set of activation systems (section 4.6.1.4). Since each taxonomy considers knowledge generated by activity steps or rather activation steps from the perspective of the content-related connection of corresponding sequences (activity or activation steps), any Knowledge overview depicts the composition of by activities induced or by activations induced knowledge with the aid of the taxonomic ordering of corresponding model elements within a certain hierarchy. Because of the strictly formalized hierarchical relation of systems specified, by activities induced or by activations induced knowledge can be considered on both, the knowledge-management perspective and the ANN perspective. Since different forms of hierarchies can be established on base of the same modeling objects, the following speaks rather from a taxonomy than from a hierarchy. Hence, we have

$$\begin{aligned} \text{KnowledgeOverView} = \\ \{G_{AandNAKOV_1}, G_{AandNAKOV_2}, G_{AandNAKOV_}, G_{AandNAKOV_{nKOV}}\}, \quad (4.65) \\ \text{with } \text{KnowledgeOverView} \neq \emptyset. \end{aligned}$$

KnowledgeOverview as Graph: According to the graph-oriented definition of Eq. 4.20, the Knowledge overview establishes the model characterization of $AandNAKOV_i$, which stands for the activation system level A and the neuronal activation system level NA taking the Knowledge overview KOV and we set up for any Knowledge view $i = 1, \dots, nKOV$ the graph

$$G_{AandNAKOV} = (V_{AandNAKOV}, E_{AandNAKOV}), \text{ with}$$

$V_{AandNAKOV} \supseteq \{KnowledgeTaxonomyVertices, P, R, Neurons, SB\}$,
and

$$E_{AandNAKOV} \supseteq \{KnowledgeTaxonomyEdges, Z, SBR\}. \quad (4.66)$$

Please consider here that vertex sets and edge sets are considered as super sets as they represent the theoretically conceivable manifestation of modeling objects. Limitations, as presented by the detailed presentation of its subsets in the following, therefore can be interpreted as syntax of the Knowledge overview.

The Knowledge graph vertices $V_{AandNAKOV}$ are further detailed as follows: Since each modeling object j of the vertex sets J can be considered either as input object $j \in V_{AandNAKOV_{\mathcal{R}}}$, as inner object $j \in V_{AandNAKOV_{\mathcal{V}_i}}$ or as output object $j \in V_{AandNAKOV_{\mathcal{V}_o}}$, for each set specified in the following it holds $J = J_{\mathcal{R}} \cup J_{\mathcal{V}_i} \cup J_{\mathcal{V}_o}$. Its detailed set construction and syntax can be found at section B.21.

The knowledge graph edges $E_{AandNAKOV}$ are further detailed as follows: Since each modeling object j of the edge sets J can be considered either as input object $j \in E_{AandNAKOV_{\mathcal{R}}}$, as inner object $j \in E_{AandNAKOV_{\mathcal{V}_i}}$ or as output object $j \in E_{AandNAKOV_{\mathcal{V}_o}}$, for each set specified in the following it holds $J = J_{\mathcal{R}} \cup J_{\mathcal{V}_i} \cup J_{\mathcal{V}_o}$. Its detailed set construction and syntax can be found at section B.22.

KnowledgeOverview as Model: In order to satisfy the dynamic model definition of discrete systems of Eq. 2.5, and considering the entities defined, the KnowledgeOverview is specified as model $\mathcal{V}_{AandNAKOV}^{(k)}$ so that temporal effects can be considered when desired:

$$\begin{aligned} \mathcal{V}_{AandNAKOV}^{(k+1)} &= f(\mathcal{R}_{AandNAKOV}^{(k)}, \mathcal{V}_{AandNAKOV}^{(k)}), \text{ with} \\ \mathcal{R}_{AandNAKOV}^{(k)} \text{ and } \mathcal{V}_{AandNAKOV}^{(k)} &\text{ on base of Eq. 4.66} \\ \text{following the model-oriented, graph-based} \\ \text{business process description of Eq. 4.19.} \end{aligned} \quad (4.67)$$

The KnowledgeOverview model elements $\mathcal{V}_{AandNAKOV}^{(k)}$ therefore consider the Knowledge graph vertices and edges defined.

Outer effects on the Knowledge Overview model $\mathcal{R}_{AandNAKOV}^{(k)}$ can issue the very same graph elements by principle, too. According to the activity system designed in section 4.6.1.3, and according to the neuronal activation system designed in sec-

tion 4.6.1.4, these either must be required by the task environment or by the super activity environment as a kind of preceding requirement for the model established, so that $\mathcal{R}_{AandNAKOV}^{(k)}$ follows $\mathcal{R}_{A_{ANN}}^{(k)}$, or alternatively, these must be required by the activity environment or by the super activation environment as a kind of preceding requirement for the model established, so that $\mathcal{R}_{AandNAKOV}^{(k)}$ follows $\mathcal{R}_{NA_{ANN}}^{(k)}$. Please remark here that this goes beyond the isolated view on an individual activity-based or activation-based knowledge generation constructed by the flow-oriented knowledge view, which issues only the question what activity-induced or activation-induced knowledge is generated by preceding activities or activations and disregards the question from which activity or activation this input is generated.

Similar to the determination of a graph vertex or edge to be a model input or an inner model object, the determination of an object to be a model output is carried out: Only if modeling objects are directly considered as model input for consecutively affected activities and activations (see deliverables of corresponding *KnowledgeViews*, the latter kind of knowledge objects, information objects and mixture objects are considered to be a model output of the selected activities and activations. Hence, any kind of consecutively affected activity-based or activation-based knowledge generation is considered as part of the model itself and an overview over the current activity-based or activation-based knowledge generation is set up. Please remark here that this goes beyond the isolated view on an individual activity-based or activation-based knowledge production constructed by the corresponding flow-oriented view, which issues only the question what activity-based or activation-based output is generated and disregards the question for which activity or activation this output is generated.

The function $f(\dots)$ refers to the highlighting of elements within the Knowledge overview model constructed and follows the activation of the neuronal system of individual activities and activations, so that the knowledge view state is transferred from $\mathcal{V}_{AandNAKOV}^{(k)}$ to $\mathcal{V}_{AandNAKOV}^{(k+1)}$ and follows the corresponding transfers of the ANN system state from $\mathcal{V}_{A_{ANN}}^{(k)}$ to $\mathcal{V}_{A_{ANN}}^{(k+1)}$ and from $\mathcal{V}_{NA_{ANN}}^{(k)}$ to $\mathcal{V}_{NA_{ANN}}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment of ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design. For example, by this, the activation of a current knowledge thread can be indicated by coloring the corresponding activity-induced knowledge vertex or activation-induced knowledge vertex.

KnowledgeOverview Conventions: For the KnowledgeOverview of a first version, CoNM-modeling-language-based Knowledge overview models are

constructed by the following design rules as they have been summarized in section C.10.

KnowledgeOverview Manifestations: The Knowledge Overview obtains different kinds of manifestations, all of which follow the syntax specified above. Here, we have the *Taxonomic KnowledgeObject OverView* providing for any activity-induced or activation-induced object i the graph $G_{AandNA_{KOOV}}$, that issues the entire collection of knowledge objects:

$$\begin{aligned}
 & KnowledgeObjectOverView = \\
 & \{G_{AandNA_{KOOV_1}}, G_{AandNA_{KOOV_2}}, G_{AandNA_{KOOV_{...}}}, G_{AandNA_{KOOV_{n_{KOOV}}}}\}, \\
 & \text{with } G_{AandNA_{KOOV}} = (V_{AandNA_{KOOV}}, E_{AandNA_{KOOV}}) = G_{AandNA_{KOOV}}, \\
 & \quad \text{having } TacitKnowledgeTaxonomyVertices \neq \emptyset, \\
 & \quad \text{and } ExplicitKnowledgeTaxonomyVertices = \emptyset.
 \end{aligned} \tag{4.68}$$

The corresponding model refers to $\mathcal{V}_{AandNA_{KOOV}}^{(k+1)} = f(\mathcal{R}_{AandNA_{KOOV}}^{(k)})$, $\mathcal{V}_{AandNA_{KOOV}}^{(k)}$.

Further, we have the *TaxonomicInformationObject OverView* providing for any activity-induced or activation-induced object i the graph $G_{AandNA_{IOOV}}$, that issues the entire collection of information objects:

$$\begin{aligned}
 & InformationObjectOverView = \\
 & \{G_{AandNA_{IOOV_1}}, G_{AandNA_{IOOV_2}}, G_{AandNA_{IOOV_{...}}}, G_{AandNA_{IOOV_{n_{IOOV}}}}\}, \\
 & \text{with } G_{AandNA_{IOOV}} = (V_{AandNA_{IOOV}}, E_{AandNA_{IOOV}}) = G_{AandNA_{IOOV}}, \\
 & \quad \text{having } TacitKnowledgeTaxonomyVertices = \emptyset, \\
 & \quad \text{and } ExplicitKnowledgeTaxonomyVertices \neq \emptyset.
 \end{aligned} \tag{4.69}$$

The corresponding model refers to $\mathcal{V}_{AandNA_{IOOV}}^{(k+1)} = f(\mathcal{R}_{AandNA_{IOOV}}^{(k)})$, $\mathcal{V}_{AandNA_{IOOV}}^{(k)}$.

Hence, we have the KnowledgeOverview, that units the taxonomic collection of Knowledge views and represents more than the individual views.

4.6.3.7 Organizational Views

Based on the CoNM meta-model designed in section 4.1.3, the CoNM modeling language follows the organizational perspective definition presented (Def. 39). In

regard to Fig. 4.14, and following the generic view specification of section 4.6.2, the Organizational perspective is characterized by the following kinds of views. First, organization views, and second, organizational overviews. Each of them is specified in the following.

1) Organization Views

The organization views on the NEM system are established by the selection of an atomic or non-atomic process out of the set of process systems (section 4.6.1.2). Since each taxonomy considers organizational objects by activity steps from the perspective of the content-related connection of corresponding activity sequences (process steps), the organization view of each process selection depicts the precise task processing sequence as well as any task alternatives. Because of the strictly formalized hierarchical relation of systems specified, activities can be considered from both, the knowledge-management perspective and the ANN perspective.

Since the organization view is similar to the process view, but having the focus on organizational object distribution, this includes its generation and use in tasks or rather sub-processes, the following principally provides the same syntax than the syntax presented in section 4.6.3.3 (process views) and establishes a by-content differentiated view. So, for example all the tasks and sub-processes are considered that distribute and reuse a certain organizational object. Hence, we have

$$\begin{aligned} \text{OrganizationView} = \\ \{G_{P_{OV_1}}, G_{P_{OV_2}}, G_{P_{OV...}}, G_{P_{OV_{n_{OV}}}}\} \quad (4.70) \\ \text{with } \text{OrganizationView} \neq \emptyset. \end{aligned}$$

OrganizationView as Graph: According to the graph-oriented definition of Eq. 4.20, the organization view establishes the model characterization of P_{OV_i} , which stands for the process system level P taking the organization view OV and for any task or rather sub-process $i = 1, \dots, n_{OV}$ we set up

$$\begin{aligned} G_{P_{OV}} = (V_{P_{OV}}, E_{P_{OV}}), \\ \text{with} \\ V_{P_{OV}} \text{ and } E_{P_{OV}} \quad (4.71) \\ \text{on base of Eq. 4.25 for process views.} \end{aligned}$$

Please consider here that vertex sets and edge sets therefore follow the same syntax as well as graph vertex and graph edge element definitions as were presented in the corresponding sections.

OrganizationView as Model: In order to satisfy the dynamic model definition of discrete systems of Eq. 2.5, and considering the entities defined, the OrganizationView is specified as model $\mathcal{V}_{P_{OV}}^{(k)}$ so that temporal effects can be considered when desired:

$$\begin{aligned} \mathcal{V}_{P_{OV}}^{(k+1)} &= f(\mathcal{R}_{P_{OV}}^{(k)}, \mathcal{V}_{P_{OV}}^{(k)}), \text{ with} \\ \mathcal{R}_{P_{OV}}^{(k)} \text{ and } \mathcal{V}_{P_{OV}}^{(k)} &\text{ on base of Eq. 4.71} \end{aligned} \quad (4.72)$$

following the model-oriented, graph-based
business process description of Eq. 4.19.

The OrganizationView model elements $\mathcal{V}_{P_{OV}}^{(k)}$ therefore consider the organization graph vertices and edges defined.

Outer effects on the OrganizationView model $\mathcal{R}_{P_{OV}}^{(k)}$ can issue the very same graph elements by principle, too. According to the process system designed in section 4.6.1.2, these either must be required by the network environment or by the super process environment as a kind of preceding requirement for the model established, so that $\mathcal{R}_{P_{OV}}^{(k)}$ follows $\mathcal{R}_{P_{ANN}}^{(k)}$.

Similar to the determination of a graph vertex or edge to be a model input or an inner model object, the determination of an object to be a model output is carried out: Only if modeling objects are directly considered as model input for consecutively affected processes, they are considered to be a model output of the selected process. In case organization object outcomes are delivered for a process stakeholder, such as the customer, these are considered as *deliverables*. Following the CoNM meta-model, then, non-tangible resources, such as an oral presentation or knowledge generated, are considered as *service* modeling item. Further, tangible resources, such as a document produced, data generated or components manufactured, are considered as *product*. Hence, these are considered as a kind of enterprise output, department output, or any other kind of ontological level's output.

The function $f(\dots)$ refers to the highlighting of elements within the organization view model constructed and follows the activation of the neuronal system of individual processes, so that the organization view state is transferred from $\mathcal{V}_{P_{OV}}^{(k)}$ to $\mathcal{V}_{P_{OV}}^{(k+1)}$ and follows the corresponding transfers of the ANN system state from $\mathcal{V}_{P_{ANN}}^{(k)}$ to $\mathcal{V}_{P_{ANN}}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment

ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design. For example, by this, the activation of a current process thread can be indicated by coloring the corresponding control flow edge.

OrganizationView Conventions: For the OrganizationView of a first version, CoNM-modeling-language-based organization view models are constructed by the same design rules than the corresponding process views.

OrganizationView Manifestations: The OrganizationView obtains different kinds of manifestations, all of which follow the syntax specified above and consider the generic views specified in section 4.6.2. Hence, we have an OrganizationView in VR, that presents the tempo-spatial positioning of process-induced and task-induced organization objects within the virtual reality in 3D. We have an OrganizationView in AR, that presents the tempo-spatial positioning of process-induced and task-induced organization objects as augmentation within the real world in 3D. And we have the OrganizationView in 2D, that presents the tempo-spatial positioning of process-induced and task-induced organization objects from two defined perspectives (ground plan and sketch plan) as well as its reading optimized visualization.

2) Organizational Overviews

The organizational overviews on the NEM system are established by the selection of an atomic or non-atomic neuronal activity out of the set of activation systems (section 4.6.1.4). Since each taxonomy considers neuronal interpretations of modeling objects from the perspective of the content-related connection of activation sequences (activation steps), any Organizational overview depicts the composition of neuronal activities with the aid of the taxonomic ordering of corresponding model elements within a certain hierarchy. Due to the strictly formalized hierarchical relation of systems specified, activations can be considered on both, the knowledge-management perspective and the ANN perspective. Since different forms of hierarchies can be established on base of the same modeling objects, the following speaks rather from a taxonomy than from a hierarchy. Hence, we have

$$\begin{aligned} \text{OrganizationalOverView} = \\ \{G_{NAOOV_1}, G_{NAOOV_2}, G_{NAOOV_{\dots}}, G_{NAOOV_{nOOV}}\}, \end{aligned} \quad (4.73)$$

with $\text{OrganizationalOverView} \neq \emptyset$.

OrganizationalOverview as Graph: According to the graph-oriented definition of Eq. 4.20, the Organizational overview establishes the model characterization

of NA_{OOV_i} , which stands for the neuronal activation system level NA taking the Organizational overview OOV and we set up for any organizational object $i = 1, \dots, n_{OOV}$

$$G_{NA_{OOV}} = (V_{NA_{OOV}}, E_{NA_{OOV}}), \text{ with}$$

$$V_{NA_{OOV}} \supseteq \{\text{OrganizationalTaxonomyVertices}, SB\}, \quad (4.74)$$

and

$$E_{NA_{OOV}} \supseteq \{\text{OrganizationalTaxonomyEdges}, SBR\}.$$

Please consider here that vertex sets and edge sets are considered as super sets as they represent the theoretically conceivable manifestation of modeling objects. Limitations, as presented by the detailed presentation of its subsets in the following, therefore can be interpreted as syntax of the Organizational overview.

The organizational graph vertices $V_{NA_{OOV}}$ are further detailed as follows: Since each modeling object j of the vertex sets J can be considered either as input object $j \in V_{NA_{OOV_R}}$, as inner object $j \in V_{NA_{OOV_{V_i}}}$ or as output object $j \in V_{NA_{OOV_{V_o}}}$, for each set specified in the following it holds $J = J_R \cup J_{V_i} \cup J_{V_o}$. Its detailed set construction and syntax can be found at section B.23.

The organization graph edges $E_{NA_{OOV}}$ are further detailed as follows: Since each modeling object j of the edge sets J can be considered either as input object $j \in E_{NA_{OOV_R}}$, as inner object $j \in E_{NA_{OOV_{V_i}}}$ or as output object $j \in E_{NA_{OOV_{V_o}}}$, for each set specified in the following it holds $J = J_R \cup J_{V_i} \cup J_{V_o}$. Its detailed set construction and syntax can be found at section B.24.

OrganizationalOverview as Model: In order to satisfy the dynamic model definition of discrete systems of Eq. 2.5, and considering the entities defined, the OrganizationalOverview is specified as model $\mathcal{V}_{NA_{OOV}}^{(k)}$ so that temporal effects can be considered when desired:

$$\mathcal{V}_{NA_{OOV}}^{(k+1)} = f(\mathcal{R}_{NA_{OOV}}^{(k)}, \mathcal{V}_{NA_{OOV}}^{(k)}), \text{ with}$$

$$\mathcal{R}_{NA_{OOV}}^{(k)} \text{ and } \mathcal{V}_{NA_{OOV}}^{(k)} \text{ on base of Eq. 4.74} \quad (4.75)$$

following the model-oriented, graph-based
business process description of Eq. 4.19.

The OrganizationalOverview model elements $\mathcal{V}_{NA_{OOV}}^{(k)}$ therefore consider the Organizational graph vertices and edges defined.

Outer effects on the OrganizationalOverview model $\mathcal{R}_{NA_{OOV}}^{(k)}$ can issue the very same graph elements by principle, too. According to the neuronal activation system designed in section 4.6.1.4, these either must be required by the activity environment or by the super activation environment as a kind of preceding requirement for the model established, so that $\mathcal{R}_{NA_{OOV}}^{(k)}$ follows $\mathcal{R}_{NA_{ANN}}^{(k)}$. Please remark here that this goes beyond the isolated view on an individual organization-specific activation constructed by the flow-oriented organization view, which issues only the question what organization-object-specific activation input is generated by preceding activities or activations and disregards the question from which organization-object-specific activation this input is generated.

Similar to the determination of a graph vertex or edge to be a model input or an inner model object, the determination of an object to be a model output is carried out: Only if modeling objects are directly considered as model input for consecutively affected organizational objects (see deliverables of corresponding *OrganizationViews*), the latter kind of activities are considered to be a model output of the selected organization-object-specific activities. Hence, any kind of consecutively affected organization-object-specific activation is considered as part of the model itself and an overview over the current organization-object-specific activation is set up. Please remark here that this goes beyond the isolated view on an individual organization-object-specific activation constructed by the flow-oriented organization view, which issues only the question what activity or activation output is generated and disregards the question for which organization-object-specific activation this output is generated.

The function $f(\dots)$ refers to the highlighting of elements within the Organizational overview model constructed and follows the activation of the neuronal system of individual activations, so that the activation view state is transferred from $\mathcal{V}_{NA_{OOV}}^{(k)}$ to $\mathcal{V}_{NA_{OOV}}^{(k+1)}$ and follows the transfer of the ANN system state from $\mathcal{V}_{NA_{ANN}}^{(k)}$ to $\mathcal{V}_{NA_{ANN}}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment of ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design. For example, by this, the activation of a current organization object thread can be indicated by coloring the corresponding activity vertex or activation vertex.

OrganizationalOverview Conventions: For the OrganizationalOverview of a first version, CoNM-modeling-language-based Organizational overview models are constructed by the following design rules as they have been summarized in section C.11.

OrganizationalOverview Manifestations: The Organizational Overview obtains different kinds of manifestations, all of which follow the syntax specified above. First, we have the *TaxonomicUnitOverView* providing for any organizational activation object i the graph G_{NAUOV} , that issues the entire collection of unit-specific activations:

$$\begin{aligned}
 & \text{UnitOverView} = \\
 & \{G_{NAUOV_1}, G_{NAUOV_2}, G_{NAUOV_{\dots}}, G_{NAUOV_{n_{UOV}}}\}, \\
 & \text{with } G_{NAUOV} = (V_{NAUOV}, E_{NAUOV}) = G_{NAOOV}, \\
 & \quad \text{having } \text{UnitTaxonomyVertices} \neq \emptyset, \\
 & \quad \text{RoleTaxonomyVertices} = \emptyset, \\
 & \quad \text{CPSTaxonomyVertices} = \emptyset, \\
 & \quad \text{PeopleTaxonomyVertices} = \emptyset, \\
 & \quad \text{and ActivationTaxonomyVertices} = \emptyset.
 \end{aligned} \tag{4.76}$$

The corresponding model refers to $\mathcal{V}_{NAUOV}^{(k+1)} = f(\mathcal{R}_{NAUOV}^{(k)}, \mathcal{V}_{NAUOV}^{(k)})$.

Second, we have the *TaxonomicRoleOverView* providing for any organizational activation object i the graph G_{NAROV} , that issues the entire collection of information-system-specific activations:

$$\begin{aligned}
 & \text{RoleOverView} = \\
 & \{G_{NAROV_1}, G_{NAROV_2}, G_{NAROV_{\dots}}, G_{NAROV_{n_{ROV}}}\}, \\
 & \text{with } G_{NAROV} = (V_{NAROV}, E_{NAROV}) = G_{NAOOV}, \\
 & \quad \text{having } \text{UnitTaxonomyVertices} = \emptyset, \\
 & \quad \text{RoleTaxonomyVertices} \neq \emptyset, \\
 & \quad \text{CPSTaxonomyVertices} = \emptyset, \\
 & \quad \text{PeopleTaxonomyVertices} = \emptyset, \\
 & \quad \text{and ActivationTaxonomyVertices} = \emptyset.
 \end{aligned} \tag{4.77}$$

The corresponding model refers to $\mathcal{V}_{NAROV}^{(k+1)} = f(\mathcal{R}_{NAROV}^{(k)}, \mathcal{V}_{NAROV}^{(k)})$.

Third, we have the *TaxonomicCPSOverView* providing for any organizational activation object i the graph $G_{NACPSOV}$, that issues the entire collection of CPS-specific activations:

$$\begin{aligned}
& CPSOverView = \\
& \{G_{NA_{CPSOV_1}}, G_{NA_{CPSOV_2}}, G_{NA_{CPSOV_{...}}}, G_{NA_{CPSOV_{n_{CPSOV}}}}\}, \\
& \text{with } G_{NA_{CPSOV}} = (V_{NA_{CPSOV}}, E_{NA_{CPSOV}}) = G_{NA_{OOV}}, \\
& \quad \text{having } UnitTaxonomyVertices = \emptyset, \\
& \quad RoleTaxonomyVertices = \emptyset, \\
& \quad CPSTaxonomyVertices \neq \emptyset, \\
& \quad PeopleTaxonomyVertices = \emptyset, \\
& \quad \text{and } ActivationTaxonomyVertices = \emptyset.
\end{aligned} \tag{4.78}$$

The corresponding model refers to $\mathcal{V}_{NA_{CPSOV}}^{(k+1)} = f(\mathcal{R}_{NA_{CPSOV}}^{(k)}, \mathcal{V}_{NA_{CPSOV}}^{(k)})$.

Fourth, we have the *Taxonomic PeopleOverView* providing for any organizational activation object i the graph $G_{NA_{POV}}$, that issues the entire collection of people-specific activations:

$$\begin{aligned}
& PeopleOverView = \\
& \{G_{NA_{POV_1}}, G_{NA_{POV_2}}, G_{NA_{POV_{...}}}, G_{NA_{POV_{n_{POV}}}}\}, \\
& \text{with } G_{NA_{POV}} = (V_{NA_{POV}}, E_{NA_{POV}}) = G_{NA_{OOV}}, \\
& \quad \text{having } UnitTaxonomyVertices = \emptyset, \\
& \quad RoleTaxonomyVertices = \emptyset, \\
& \quad CPSTaxonomyVertices = \emptyset, \\
& \quad PeopleTaxonomyVertices \neq \emptyset, \\
& \quad \text{and } ActivationTaxonomyVertices = \emptyset.
\end{aligned} \tag{4.79}$$

The corresponding model refers to $\mathcal{V}_{NA_{POV}}^{(k+1)} = f(\mathcal{R}_{NA_{POV}}^{(k)}, \mathcal{V}_{NA_{POV}}^{(k)})$.

In principle, further the *ActivationOverView* can be established. Since this focuses on the neurons of the system, which is not the kind of perspective intended to be established, here, this view is defined in Eq. 4.58 of the section addressing the Process ANN perspective.

Hence, we have the *OrganizationalOverview*, that units the taxonomic collection of Organizational views and represents more than the individual views.

4.6.3.8 Informational Views

Based on the CoNM meta-model designed in section 4.1.3, the CoNM modeling language follows the informational perspective definition presented (Def. 37). In

regard with Fig. 4.14, and following the generic view specification of section 4.6.2, the Information perspective is characterized by the following kinds of views. First, information views, and second, informational overviews. Each of them is specified in the following.

1) Information Views

The information views on the NEM system are established by the selection of an atomic or non-atomic process out of the set of process systems (section 4.6.1.2). Since each taxonomy considers informational objects by activity steps from the perspective of the content-related connection of corresponding activity sequences (process steps), the information view of each process selection depicts the precise task processing sequence as well as any task alternatives. Due to the strictly formalized hierarchical relation of systems specified, activities can be considered from both, the knowledge-management perspective and the ANN perspective.

Since the information view is similar to the process view, but having the focus on informational object distribution, this includes its generation and use in tasks or rather sub-processes, the following principally provides the same syntax than the syntax presented in section 4.6.3.3 (process views) and establishes a by-content differentiated view. So, for example, all the tasks and sub-processes are considered that distribute and reuse a certain informational object. Hence, we have

$$\begin{aligned} \text{InformationView} = \\ \{G_{P_{IV_1}}, G_{P_{IV_2}}, G_{P_{IV_{...}}}, G_{P_{IV_{n_{IV}}}}\} \\ \text{with } \text{InformationView} \neq \emptyset. \end{aligned} \quad (4.80)$$

Information View as Graph: According to the graph-oriented definition of Eq. 4.20, the information view establishes the model characterization of P_{IV_i} , which stands for the process system level P taking the information view IV and for any task or rather sub-process $i = 1, \dots, n_{IV}$ we set up

$$\begin{aligned} G_{P_{IV}} = (V_{P_{IV}}, E_{P_{IV}}), \\ \text{with} \\ V_{P_{IV}} \text{ and } E_{P_{IV}} \\ \text{on base of Eq. 4.25 for process views.} \end{aligned} \quad (4.81)$$

Please consider here that vertex sets and edge sets therefore follow the same syntax as well as graph vertex and graph edge element definitions as were presented in the corresponding sections.

InformationView as Model: In order to satisfy the dynamic model definition of discrete systems of Eq. 2.5, and considering the entities defined, the InformationView is specified as model $\mathcal{V}_{P_{IV}}^{(k)}$ so that temporal effects can be considered when desired:

$$\begin{aligned} \mathcal{V}_{P_{IV}}^{(k+1)} &= f(\mathcal{R}_{P_{IV}}^{(k)}, \mathcal{V}_{P_{IV}}^{(k)}), \text{ with} \\ \mathcal{R}_{P_{IV}}^{(k)} \text{ and } \mathcal{V}_{P_{IV}}^{(k)} &\text{ on base of Eq. 4.81} \end{aligned} \quad (4.82)$$

following the model-oriented, graph-based
business process description of Eq. 4.19.

The InformationView model elements $\mathcal{V}_{P_{IV}}^{(k)}$ therefore consider the information graph vertices and edges defined.

Outer effects on the InformationView model $\mathcal{R}_{P_{IV}}^{(k)}$ can issue the very same graph elements by principle, too. According to the process system designed in section 4.6.1.2, these either must be required by the network environment or by the super process environment as a kind of preceding requirement for the model established, so that $\mathcal{R}_{P_{IV}}^{(k)}$ follows $\mathcal{R}_{P_{ANN}}^{(k)}$.

Similar to the determination of a graph vertex or edge to be a model input or an inner model object, the determination of an object to be a model output is carried out: Only if modeling objects are directly considered as model input for consecutively affected processes, they are considered to be a model output of the selected process. In case information object outcomes are delivered for a process stakeholder, such as the customer, these are considered as *deliverables*. Following the CoNM meta-model, then, non-tangible resources, such as an oral presentation or knowledge generated, are considered as *service* modeling item. Further, tangible resources, such as a document produced, data generated or components manufactured, are considered as *product*. Hence, these are considered as a kind of enterprise output, department output, or any other kind of ontological level's output.

The function $f(\dots)$ refers to the highlighting of elements within the information view model constructed and follows the activation of the neuronal system of individual processes, so that the information view state is transferred from $\mathcal{V}_{P_{IV}}^{(k)}$ to $\mathcal{V}_{P_{IV}}^{(k+1)}$ and follows the corresponding transfers of the ANN system state from $\mathcal{V}_{P_{ANN}}^{(k)}$ to $\mathcal{V}_{P_{ANN}}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment

ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design. For example, by this, the activation of a current process thread can be indicated by coloring the corresponding control flow edge.

InformationView Conventions: For the InformationView of a first version, CoNM-modeling-language-based information view models are constructed by the same design rules than the corresponding process views.

InformationView Manifestations: The InformationView obtains different kinds of manifestations, all of which follow the syntax specified above and consider the generic views specified in section 4.6.2. Hence, we have an InformationView in VR, that presents the tempo-spatial positioning of process-induced and task-induced information objects within the virtual reality in 3D. We have an InformationView in AR, that presents the tempo-spatial positioning of process-induced and task-induced information objects as augmentation within the real world in 3D. And we have the InformationView in 2D, that presents the tempo-spatial positioning of process-induced and task-induced information objects from two defined perspectives (ground plan and sketch plan) as well as its reading optimized visualization.

2) Informational Overviews

The informational overviews on the NEM system are established by the selection of an atomic or non-atomic neuronal activity out of the set of activation systems (section 4.6.1.4). Since each taxonomy considers neuronal interpretations of modeling objects from the perspective of the content-related connection of activation sequences (activation steps), any Informational overview depicts the composition of neuronal activities with the aid of the taxonomic ordering of corresponding model elements within a certain hierarchy. Due to the strictly formalized hierarchical relation of systems specified, activations can be considered on both, the knowledge-management perspective and the ANN perspective. Since different forms of hierarchies can be established on base of the same modeling objects, the following speaks rather from a taxonomy than from a hierarchy. Hence, we have

$$\begin{aligned} \text{InformationalOverView} = \\ \{G_{NA_{IOV_1}}, G_{NA_{IOV_2}}, G_{NA_{IOV...}}, G_{NA_{IOV_{n_{IOV}}}}\}, \\ \text{with } \text{InformationalOverView} \neq \emptyset. \end{aligned} \quad (4.83)$$

InformationalOverview as Graph: According to the graph-oriented definition of Eq. 4.20, the Informational overview establishes the model characterization of

NA_{IOV_i} , which stands for the neuronal activation system level NA taking the Informational overview IOV and we set up for any informational object $i = 1, \dots, n_{IOV}$

$$G_{NA_{IOV}} = (V_{NA_{IOV}}, E_{NA_{IOV}}),$$

with

$$V_{NA_{IOV}} \supseteq \{InformationalTaxonomyVertices, SB\}, \quad (4.84)$$

and

$$E_{NA_{IOV}} \supseteq \{InformationalTaxonomyEdges, SBR\}.$$

Please consider here that vertex sets and edge sets are considered as super sets as they represent the theoretically conceivable manifestation of modeling objects. Limitations, as presented by the detailed presentation of its subsets in the following, therefore can be interpreted as syntax of the Informational overview.

The Informational graph vertices $V_{NA_{IOV}}$ are further detailed as follows: Since each modeling object j of the vertex sets J can be considered either as input object $j \in V_{NA_{IOV_R}}$, as inner object $j \in V_{NA_{IOV_{V_i}}}$ or as output object $j \in V_{NA_{IOV_{V_o}}}$, for each set specified in the following it holds $J = J_R \cup J_{V_i} \cup J_{V_o}$. Its detailed set construction and syntax can be found at section B.25.

The information graph edges $E_{NA_{IOV}}$ are further detailed as follows: Since each modeling object j of the edge sets J can be considered either as input object $j \in E_{NA_{IOV_R}}$, as inner object $j \in E_{NA_{IOV_{V_i}}}$ or as output object $j \in E_{NA_{IOV_{V_o}}}$, for each set specified in the following it holds $J = J_R \cup J_{V_i} \cup J_{V_o}$. Its detailed set construction and syntax can be found at section B.26.

InformationalOverview as Model: In order to satisfy the dynamic model definition of discrete systems of Eq. 2.5, and considering the entities defined, the InformationalOverview is specified as model $\mathcal{V}_{NA_{IOV}}^{(k)}$ so that temporal effects can be considered when desired:

$$\begin{aligned} \mathcal{V}_{NA_{IOV}}^{(k+1)} &= f(\mathcal{R}_{NA_{IOV}}^{(k)}, \mathcal{V}_{NA_{IOV}}^{(k)}), \text{ with} \\ \mathcal{R}_{NA_{IOV}}^{(k)} \text{ and } \mathcal{V}_{NA_{IOV}}^{(k)} &\text{ on base of Eq. 4.84} \\ \text{following the model-oriented, graph-based} \\ \text{business process description of Eq. 4.19.} \end{aligned} \quad (4.85)$$

The InformationalOverview model elements $\mathcal{V}_{NA_{IOV}}^{(k)}$ therefore consider the Informational graph vertices and edges defined.

Outer effects on the InformationalOverview model $\mathcal{R}_{NA_{IOV}}^{(k)}$ can issue the very same graph elements by principle, too. According to the neuronal activation system designed in section 4.6.1.4, these either must be required by the activity environment or by the super activation environment as a kind of preceding requirement for the model established, so that $\mathcal{R}_{NA_{IOV}}^{(k)}$ follows $\mathcal{R}_{NA_{ANN}}^{(k)}$. Please remark here that this goes beyond the isolated view on an individual activity or activation constructed by the flow-oriented information view, which issues only the question what information-object-specific activation input is generated by preceding activities or activations and disregards the question from which information-object-specific activation this input is generated.

Similar to the determination of a graph vertex or edge to be a model input or an inner model object, the determination of an object to be a model output is carried out: Only if modeling objects are directly considered as model input for consecutively affected informational objects (see deliverables of corresponding *InformationViews*), the latter kind of activities are considered to be a model output of the selected information-object-specific activities. Hence, any kind of consecutively affected information-object-specific activation is considered as part of the model itself and an overview over the current information-object-specific activation is set up. Please remark here that this goes beyond the isolated view on an individual information-object-specific activation constructed by the flow-oriented information view, which issues only the question what activity or activation output is generated and disregards the question for which information-object-specific activation this output is generated.

The function $f(\dots)$ refers to the highlighting of elements within the Informational overview model constructed and follows the activation of the neuronal system of individual activations, so that the activation view state is transferred from $\mathcal{V}_{NA_{IOV}}^{(k)}$ to $\mathcal{V}_{NA_{IOV}}^{(k+1)}$ and follows the transfer of the ANN system state from $\mathcal{V}_{NA_{ANN}}^{(k)}$ to $\mathcal{V}_{NA_{ANN}}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment of ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design. For example, by this, the activation of a current information object thread can be indicated by coloring the corresponding activity vertex or activation vertex.

InformationalOverview Conventions: For the InformationalOverview of a first version, CoNM-modeling-language-based Informational overview models are constructed by the following design rules as they have been summarized in section C.12.

InformationalOverview Manifestations: The Informational Overview obtains different kinds of manifestations, all of which follow the syntax specified above. First, we have the *TaxonomicMachineOverView* providing for any informational activation object i the graph $G_{NA_{MOV}}$, that issues the entire collection of machine-specific activations:

$$\begin{aligned}
 & MachineOverView = \\
 & \{G_{NA_{MOV_1}}, G_{NA_{MOV_2}}, G_{NA_{MOV_...}}, G_{NA_{MOV_{n_{MOV}}}}\}, \\
 & \text{with } G_{NA_{MOV}} = (V_{NA_{MOV}}, E_{NA_{MOV}}) = G_{NA_{IOV}}, \\
 & \quad \text{having } MachineTaxonomyVertices \neq \emptyset, \\
 & \quad ISTaxonomyVertices = \emptyset, \\
 & \quad TRTaxonomyVertices = \emptyset, \\
 & \quad NTRTaxonomyVertices = \emptyset, \\
 & \quad DataTaxonomyVertices = \emptyset, \\
 & \quad \text{and } ActivationTaxonomyVertices = \emptyset.
 \end{aligned} \tag{4.86}$$

The corresponding model refers to $\mathcal{V}_{NA_{MOV}}^{(k+1)} = f(\mathcal{R}_{NA_{MOV}}^{(k)}, \mathcal{V}_{NA_{MOV}}^{(k)})$.

Second, we have the *TaxonomicInformationSystemOverView* providing for any informational activation object i the graph $G_{NA_{ISOV}}$, that issues the entire collection of information-system-specific activations:

$$\begin{aligned}
 & InformationSystemOverView = \\
 & \{G_{NA_{ISOV_1}}, G_{NA_{ISOV_2}}, G_{NA_{ISOV_...}}, G_{NA_{ISOV_{n_{ISOV}}}}\}, \\
 & \text{with } G_{NA_{ISOV}} = (V_{NA_{ISOV}}, E_{NA_{ISOV}}) = G_{NA_{IOV}}, \\
 & \quad \text{having } MachineTaxonomyVertices = \emptyset, \\
 & \quad ISTaxonomyVertices \neq \emptyset, \\
 & \quad TRTaxonomyVertices = \emptyset, \\
 & \quad NTRTaxonomyVertices = \emptyset, \\
 & \quad DataTaxonomyVertices = \emptyset, \\
 & \quad \text{and } ActivationTaxonomyVertices = \emptyset.
 \end{aligned} \tag{4.87}$$

The corresponding model refers to $\mathcal{V}_{NA_{ISOV}}^{(k+1)} = f(\mathcal{R}_{NA_{ISOV}}^{(k)}, \mathcal{V}_{NA_{ISOV}}^{(k)})$.

Third, we have the *TaxonomicTangibleResourceOverView* providing for any tangible resource activation object i the graph G_{NATROV} , that issues the entire collection of tangible-resource-specific activations:

$$\begin{aligned}
 & TangibleResourceOverView = \\
 & \{G_{NATROV_1}, G_{NATROV_2}, G_{NATROV_{\dots}}, G_{NATROV_{n_{TROV}}}\}, \\
 & \text{with } G_{NATROV} = (V_{NATROV}, E_{NATROV}) = G_{NA_{IOV}}, \\
 & \quad \text{having } MachineTaxonomyVertices = \emptyset, \\
 & \quad ISTaxonomyVertices = \emptyset, \tag{4.88} \\
 & \quad TRTaxonomyVertices \neq \emptyset, \\
 & \quad NTRTaxonomyVertices = \emptyset, \\
 & \quad DataTaxonomyVertices = \emptyset, \\
 & \text{and } ActivationTaxonomyVertices = \emptyset.
 \end{aligned}$$

The corresponding model refers to $\mathcal{V}_{NATROV}^{(k+1)} = f(\mathcal{R}_{NATROV}^{(k)}, \mathcal{V}_{NATROV}^{(k)})$.

Fourth, we have the *TaxonomicNonTangibleResourceOverView* providing for any non-tangible resource activation object i the graph $G_{NANTROV}$, that issues the entire collection of non-tangible-resource-specific activations:

$$\begin{aligned}
 & NonTangibleResourceOverView = \\
 & \{G_{NANTROV_1}, G_{NANTROV_2}, G_{NANTROV_{\dots}}, G_{NANTROV_{n_{NTR}}}\}, \\
 & \text{with } G_{NANTROV} = (V_{NANTROV}, E_{NANTROV}) = G_{NA_{IOV}}, \\
 & \quad \text{having } MachineTaxonomyVertices = \emptyset, \\
 & \quad ISTaxonomyVertices = \emptyset, \tag{4.89} \\
 & \quad TRTaxonomyVertices = \emptyset, \\
 & \quad NTRTaxonomyVertices \neq \emptyset, \\
 & \quad DataTaxonomyVertices = \emptyset, \\
 & \text{and } ActivationTaxonomyVertices = \emptyset.
 \end{aligned}$$

The corresponding model refers to $\mathcal{V}_{NANTROV}^{(k+1)} = f(\mathcal{R}_{NANTROV}^{(k)}, \mathcal{V}_{NANTROV}^{(k)})$.

Fifth, we have the *TaxonomicDataOverView* providing for any data activation object i the graph $G_{NA_{DOV}}$, that issues the entire collection of data-specific activations:

$$\begin{aligned}
DataOverView &= \{G_{NA_{DOV_1}}, G_{NA_{DOV_2}}, G_{NA_{DOV_...}}, G_{NA_{DOV_{n_{DOV}}}}\}, \\
\text{with } G_{NA_{DOV}} &= (V_{NA_{DOV}}, E_{NA_{DOV}}) = G_{NA_{IOV}}, \\
\text{having } MachineTaxonomyVertices &= \emptyset, \\
ISTaxonomyVertices &= \emptyset, \\
TRTaxonomyVertices &= \emptyset, \\
NRTaxonomyVertices &= \emptyset, \\
DataTaxonomyVertices &\neq \emptyset, \\
\text{and ActivationTaxonomyVertices} &= \emptyset.
\end{aligned} \tag{4.90}$$

The corresponding model refers to $\mathcal{V}_{NA_{DOV}}^{(k+1)} = f(\mathcal{R}_{NA_{DOV}}^{(k)}, \mathcal{V}_{NA_{DOV}}^{(k)})$.

In principle, further the *ActivationOverView* can be established. Since this focuses on the neurons of the system, which is not the kind perspective intended to be established, here, this view is defined in Eq. 4.58 of the section addressing the Process ANN perspective.

Hence, we have the *InformationalOverview*, that units the taxonomic collection of Informational views and represents more than the individual views.

4.6.3.9 Neuronal Views

Based on the CoNM meta-model designed in section 4.1.3, the CoNM modeling language follows the neuronal perspective definition presented (Def. 40). In regard to Fig. 4.14, and following the generic view specification of section 4.6.2, the Neuronal perspective is characterized by the following kinds of views. First, neuron views, and second, neuronal overviews. Each of them is specified in the following.

1) Neuron Views

The neuron views on the NEM system are established by the selection of an atomic or non-atomic activation out of the set of activation systems (section 4.6.1.4). Since each activation considers neuronal activities from the perspective of the flow of neuronal arousals sequences (arousal steps), the neuron view of each neuronal activity selection depicts the precise arousal processing sequence as well as any arousal processing alternatives. Due to the strictly formalized hierarchical relation of systems specified, neuronal activities and activations can be considered on both, the knowledge-management perspective and the ANN perspective. Hence, we have

$$\begin{aligned}
NeuronView &= \{G_{NA_{NV_1}}, G_{NA_{NV_2}}, G_{NA_{NV_...}}, G_{NA_{NV_{n_{NV}}}}\} \\
\text{with } NeuronView &\neq \emptyset.
\end{aligned} \tag{4.91}$$

NeuronView as Graph: According to the graph-oriented definition of Eq. 4.20, the neuron view establishes the model characterization of NA_{NV_i} , which stands for the neuronal activation system level NA taking the neuron view NV and for any activation $i = 1, \dots, n_{NV}$ we set up

$$G_{NA_{NV}} = (V_{NA_{NV}}, E_{NA_{NV}}),$$

with

$$\begin{aligned} V_{NA_{NV}} \supseteq & \{SB, Activations, SubActivities, KB, \\ & Neurons, AxonHills, Dendrites, NerveFibres, Synapses, \\ & AgF, AcF, AT, W, TypesArousal\}, \\ & \text{and} \\ E_{NA_{NV}} \supseteq & \{KF, Z, SBR\} \end{aligned} \quad (4.92)$$

Please consider here that vertex sets and edge sets are considered as super sets as they represent the theoretically conceivable manifestation of modeling objects. Limitations, as presented by the detailed presentation of its subsets in the following, therefore can be interpreted as syntax of the neuron view.

The neuron graph vertices $V_{NA_{NV}}$ are further detailed as follows: Since each modeling object j of the vertex sets J can be considered either as input object $j \in V_{NA_{NV_R}}$, as inner object $j \in V_{NA_{NV_{V_i}}}$ or as output object $j \in V_{NA_{NV_{V_o}}}$, for each set specified in the following it holds $J = J_R \cup J_{V_i} \cup J_{V_o}$. Its detailed set construction and syntax can be found at section B.27.

The activation graph edges $E_{NA_{NV}}$ are further detailed as follows: Since each modeling object j of the edge sets J can be considered either as input object $j \in E_{NA_{NV_R}}$, as inner object $j \in E_{NA_{NV_{V_i}}}$ or as output object $j \in E_{NA_{NV_{V_o}}}$, for each set specified in the following it holds $J = J_R \cup J_{V_i} \cup J_{V_o}$. Its detailed set construction and syntax can be found at section B.28.

NeuronView as Model: In order to satisfy the dynamic model definition of discrete systems of Eq. 2.5, and considering the entities defined, the NeuronView is specified as model $\mathcal{V}_{NA_{NV}}^{(k)}$ so that temporal effects can be considered when desired:

$$\begin{aligned} \mathcal{V}_{NA_{NV}}^{(k+1)} &= f(\mathcal{R}_{NA_{NV}}^{(k)}, \mathcal{V}_{NA_{NV}}^{(k)}), \text{ with} \\ \mathcal{R}_{NA_{NV}}^{(k)} \text{ and } \mathcal{V}_{NA_{NV}}^{(k)} &\text{ on base of Eq. 4.92} \\ \text{following the model-oriented, graph-based} \\ \text{business process description of Eq. 4.19.} \end{aligned} \quad (4.93)$$

The NeuronView model elements $\mathcal{V}_{NA_{NV}}^{(k)}$ therefore consider the neuron graph vertices and edges defined.

Outer effects on the NeuronView model $\mathcal{R}_{NA_{NV}}^{(k)}$ can issue the very same graph elements by principle, too. According to the neuronal activation system designed in section 4.6.1.4, these either must be required by the activity environment or by the super neuronal activity environment, so that $\mathcal{R}_{NA_{NV}}^{(k)}$ follows $\mathcal{R}_{NA_{ANN}}^{(k)}$.

Similar to the determination of a graph vertex or edge to be a model input or an inner model object, the determination of an object to be a model output is carried out: Only if modeling objects are directly considered as model input for consecutively affected neurons or activations, they are considered to be a model output of the selected neuronal object. In case neuronal outcomes are delivered for a process stakeholder, such as the customer, these are considered as *deliverables*. Following the CoNM meta-model, then, non-tangible resources, such as an oral presentation or knowledge generated, are considered as *service* modeling item. Further, tangible resources, such as a document produced, data generated or components manufactured, are considered as *product*. Hence, these are considered as a kind of enterprise output, department output, or any other kind of ontological level's output.

The function $f(\dots)$ refers to the highlighting of elements within the neuron view model constructed and follows the activation of the neuronal system of individual processes, so that the activation view state is transferred from $\mathcal{V}_{NA_{NV}}^{(k)}$ to $\mathcal{V}_{NA_{NV}}^{(k+1)}$ and follows the transfer of the ANN system state from $\mathcal{V}_{NA_{ANN}}^{(k)}$ to $\mathcal{V}_{NA_{ANN}}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment of ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design. For example, by this, the activation of a current neuronal object thread can be indicated by coloring the corresponding knowledge flow edge.

NeuronView Conventions: For the NeuronView of a first version, CoNM-modeling-language-based activation models are constructed by the following design rules as they have been summarized in section C.13.

NeuronView Manifestations: The NeuronView obtains different kinds of manifestations, all of which follow the syntax specified above and consider the generic views specified in section 4.6.2. Hence, we have an NeuronView in VR, that presents the tempo-spatial positioning of neurons within the virtual reality in 3D. We have an NeuronView in AR, that presents the tempo-spatial positioning of neurons as augmentation within the real world in 3D. And we have the NeuronView in 2D, that presents the tempo-spatial positioning of neurons from two defined perspectives (ground plan and sketch plan) as well as its reading optimized visualization.

2) Neuronal Overviews

The neuronal overviews on the NEM system are established by the selection of an atomic or non-atomic neuronal activity out of the set of activation systems (section 4.6.1.4). Since each taxonomy considers neuronal activity steps from the perspective of the content-related connection of neuronal arousal sequences (arousal steps), any neuronal overview depicts the composition of neurons with the aid of the taxonomic ordering of corresponding model elements within a certain hierarchy. Due to the strictly formalized hierarchical relation of systems specified, modeling objects can be considered on both, the knowledge-management perspective and the ANN perspective. Since different forms of hierarchies can be established on base of the same modeling objects, the following speaks rather from a taxonomy than from a hierarchy. Hence, we have

$$\begin{aligned} \text{NeuronalOverView} = \\ \{G_{NA_{NOV_1}}, G_{NA_{NOV_2}}, G_{NA_{NOV_..}}, G_{NA_{NOV_{n_{NOV}}}}\}, \quad (4.94) \\ \text{with } \text{NeuronalOverView} \neq \emptyset. \end{aligned}$$

NeuronalOverview as Graph: According to the graph-oriented definition of Eq. 4.20, the neuronal overview establishes the model characterization of NA_{NOV_i} , which stands for the neuronal activation system level NA taking the neuronal overview NOV and we set up for any neuronal object $i = 1, \dots, n_{NOV}$

$$\begin{aligned} G_{NA_{NOV}} = (V_{NA_{NOV}}, E_{NA_{NOV}}), \\ \text{with} \\ V_{NA_{NOV}} \supseteq \{\text{NeuronalTaxonomyVertices}, SB\}, \quad (4.95) \\ \text{and} \\ E_{NA_{NOV}} \supseteq \{\text{NeuronalTaxonomyEdges}, SBR\}. \end{aligned}$$

Please consider here that vertex sets and edge sets are considered as super sets as they represent the theoretically conceivable manifestation of modeling objects. Limitations, as presented by the detailed presentation of its subsets in the following, therefore can be interpreted as syntax of the neuronal overview.

The neuronal graph vertices $V_{NA_{NOV}}$ are further detailed as follows: Since each modeling object j of the vertex sets J can be considered either as input object $j \in V_{NA_{NOV_R}}$, as inner object $j \in V_{NA_{NOV_{V_i}}}$ or as output object $j \in V_{NA_{NOV_{V_o}}}$, for each set specified in the following it holds $J = J_R \cup J_{V_i} \cup J_{V_o}$. Its detailed set construction and syntax can be found at section B.29.

The neuronal graph edges $E_{NA_{NOV}}$ are further detailed as follows: Since each modeling object j of the edge sets J can be considered either as input object $j \in E_{NA_{NOVR}}$, as inner object $j \in E_{NA_{NOVV_i}}$ or as output object $j \in E_{NA_{NOVV_o}}$, for each set specified in the following it holds $J = J_{\mathcal{R}} \cup J_{\mathcal{V}_i} \cup J_{\mathcal{V}_o}$. Its detailed set construction and syntax can be found at section B.30.

NeuronalOverview as Model: In order to satisfy the dynamic model definition of discrete systems of Eq. 2.5, and considering the entities defined, the NeuronalOverview is specified as model $\mathcal{V}_{NA_{NOV}}^{(k)}$ so that temporal effects can be considered when desired:

$$\begin{aligned}\mathcal{V}_{NA_{NOV}}^{(k+1)} &= f(\mathcal{R}_{NA_{NOV}}^{(k)}, \mathcal{V}_{NA_{NOV}}^{(k)}), \text{ with} \\ \mathcal{R}_{NA_{NOV}}^{(k)} \text{ and } \mathcal{V}_{NA_{NOV}}^{(k)} &\text{ on base of Eq. 4.95} \\ \text{following the model-oriented, graph-based} \\ \text{business process description of Eq. 4.19.}\end{aligned}\tag{4.96}$$

The NeuronalOverview model elements $\mathcal{V}_{NA_{NOV}}^{(k)}$ therefore consider the Process ANN graph vertices and edges defined.

Outer effects on the NeuronalOverview model $\mathcal{R}_{NA_{NOV}}^{(k)}$ can issue the very same graph elements by principle, too. According to the neuronal activation system designed in section 4.6.1.4, these either must be required by the activity environment or by the super activation environment as a kind of preceding requirement for the model established, so that $\mathcal{R}_{NA_{NOV}}^{(k)}$ follows $\mathcal{R}_{NA_{ANN}}^{(k)}$. Please remark here that this goes beyond the isolated view on an individual neuron or activation constructed by the flow-oriented neuron view, which issues only the question what neuron-specific activation input is generated by preceding activities or neurons and disregards the question from which neuron or activation this input is generated.

Similar to the determination of a graph vertex or edge to be a model input or an inner model object, the determination of an object to be a model output is carried out: Only if modeling objects are directly considered as model input for consecutively affected neurons and activations (see deliverables of corresponding *NeuronViews*), the latter kind of activations are considered to be a model output of the selected neuronal objects. Hence, any kind of consecutively affected neuron or activation is considered as part of the model itself and an overview over the current neuron-specific activation is set up. Please remark here that this goes beyond the isolated view on an individual neuron or activation constructed by the flow-oriented neuron view, which issues only the question what neuron-specific activation output

is generated and disregards the question for which neuron or activation this output is generated.

The function $f(\dots)$ refers to the highlighting of elements within the Neuronal overview model constructed and follows the activation of the neuronal system of individual activations, so that the activation view state is transferred from $\mathcal{V}_{NA_{ANOV}}^{(k)}$ to $\mathcal{V}_{NA_{ANN}}^{(k+1)}$ and follows the transfer of the ANN system state from $\mathcal{V}_{NA_{ANN}}^{(k)}$ to $\mathcal{V}_{NA_{ANN}}^{(k+1)}$. Faced with various neuronal techniques for their training and establishment of ANN-specific architectures presented in section 2.5, its concrete form follows the individual ANN design. For example, by this, the activation of a current neuronal object thread can be indicated by coloring the corresponding activity vertex or activation vertex.

NeuronalOverview Conventions: For the NeuronalOverview of a first version, CoNM-modeling-language-based Process ANN overview models are constructed by the following design rules as they have been summarized in section C.14.

NeuronalOverview Manifestations: The neuronal overview obtains different kinds of manifestations, all of which follow the syntax specified above. Here, we have the *TaxonomicNeuronOverView* providing for any neuronal activation object i the graph $G_{NA_{NeOV}}$, that issues the entire collection of neurons:

$$\begin{aligned} \text{NeuronOverView} &= \{G_{NA_{NeOV_1}}, G_{NA_{NeOV_2}}, G_{NA_{NeOV_{\dots}}}, G_{NA_{NeOV_{n_{NeOV}}}}\}, \\ \text{with } G_{NA_{NeOV}} &= (V_{NA_{NeOV}}, E_{NA_{NeOV}}) = G_{NA_{NOV}}, \\ \text{having } ActivationTaxonomyVertices &= \emptyset, \\ \text{and } NeuronTaxonomyVertices &\neq \emptyset. \end{aligned} \tag{4.97}$$

The corresponding model refers to $\mathcal{V}_{NA_{NeOV}}^{(k+1)} = f(\mathcal{R}_{NA_{NeOV}}^{(k)}, \mathcal{V}_{NA_{NeOV}}^{(k)})$.

In principle, further the *ActivationOverView* can be established. Since this focuses on the Process ANN perspective on the system, which is not the kind of perspective intended to be established, here, this view is defined in Eq. 4.58 of the section addressing the Process ANN perspective.

Hence, we have the NeuronalOverview, that units the taxonomic collection of neuronal views and represents more than the individual views.

4.6.4 Modeling Shape Specification

Being faced with the syntax specified in the previous section, the hierarchy building according to a knowledge process model is satisfied and a CoNM modeling language can be specified, which is harmony with Def. 14 (knowledge process model). Jointly, with the modeling syntax, modeling convention, and the extension with modeling shapes presented in this section, the CoNM-capable modeling language is formalized and prepared for the practical application. Hence, from hereon, this modeling language is referred to as **Neuronal Modeling and Description Language (NMDL)**. The acronym is based on the following wording: First, the focus is put on neurons. Second, it is clarified that one is able to construct models as modeling items are substantiated by techniques. Third, it is made clear that one is able to individually describe modeling items, although finally, the same technical instance is characterized by different verbal denominations. Fourth, it is defined to specify a language for the CoNM, which includes syntax and conventions.

Beside the enrichment by modeling shapes, the following figures reflect the NMDL-specific, operational meta-model with whom modeling tools, such as the *Modelangelo* (section 2.6.1.4), can be set up for the modeling language use.

Faced with the omnipresent nature of some modeling items and their repeated appearance in several views, Fig. 4.17 presents the legend of the most common modeling items first. These have been derived from the meta-model designed. Further, they have devised, so that the navigation within complex hierarchical structures is simplified.

Common modeling items have been clustered by the following categories:

- Items being available at all views, the here called *general view characterization items*.
- Items issuing the *usability characterization*, with whom one is able to comfortably navigate throughout the entire collection of process models.
- Three item clusters showing items that can be found at several views.

Since items have been characterized by previous design sections, and here only a shape is presented as a complementation, the individual items are not deepened any further. Views presented in the following reuse the items specified and build on the understanding of these kinds of items.

Legend:

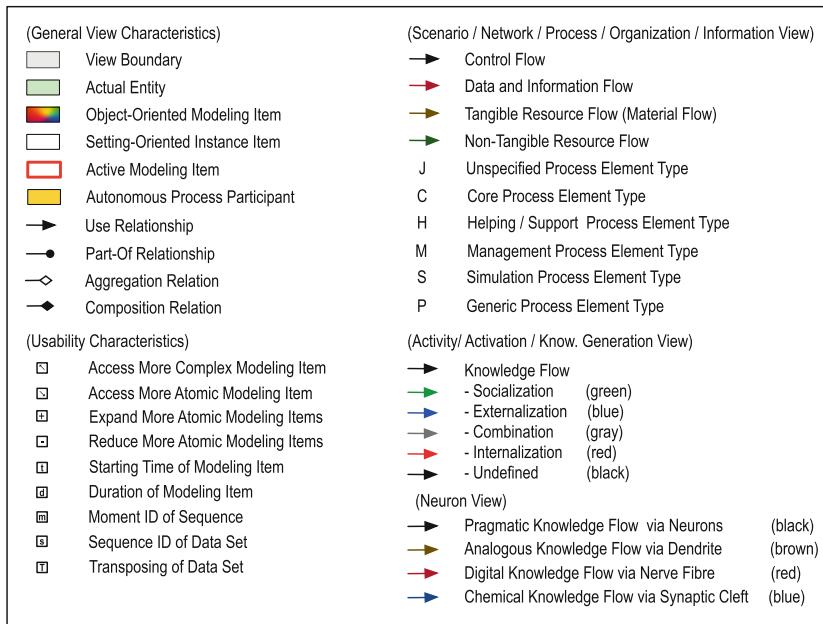
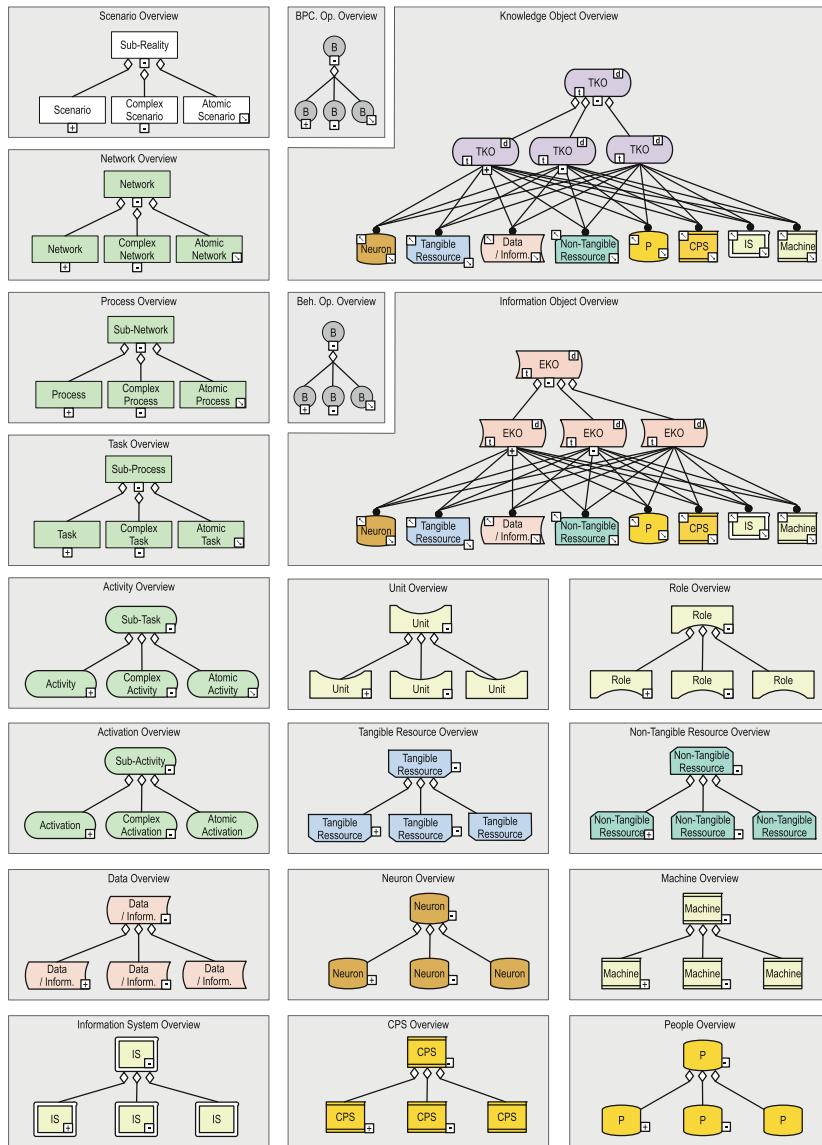


Figure 4.17 The Omnipresent Modeling Items of the NMDL (Derived From the Meta-Model Designed and Devised For Simple Navigation)

Atomic Taxonomy Views: The different taxonomic views on the knowledge process model establish the most focused view, which is in regard of a certain aspect. Fig. 4.18 presents their joint collection.

Faced with this figure, the following things become apparent:

- A remark has been put on the convenient visualization of different kinds of actual entities (according to Def. 3) all being highlighted in bright green.
- Autonomous systems have been conveniently visualized by a lush yellow.
- The two different kinds of operator views become transparent: while the *BehavioralOperatorOverview* addresses the behavioral characterization of a process system, the *BPCOperatorOverview* addresses the network interplay of process systems.

**Figure 4.18** The Atomic NMDL Taxonomy Views

- The hierarchical composition of different kinds of objects provides the same structure. By this, for all kinds of objects, either a top-down modeling approach is enabled, constructing on behalf of a decomposition of super components, or, a bottom-up modeling approach is enabled, constructing on behalf of a composition of sub-components. Of course, a hybrid modeling approach is enabled, too, considering top-down and bottom-up constructions and finally interlinking objects and object components constructed so far.

Aiming for the NKMM (section 4.5), with the aid of atomic taxonomic views of the NMDL, knowledge as well as their associated modeling objects can be accessed on either the neuronal level, or on the traditional KM level. Hence, the NMDL operationalizes knowledge-relevant entities and makes them accessible on different kinds of levels for the first time: (1) on the neuronal level, (2) the traditional KM level, and (3) the transition from the first to the latter kind of level.

Complex Taxonomy Views: The different taxonomic views on the knowledge process model establish the isolated characterization of an entire perspective, which is in regard of a certain cluster of aspects. Fig. 4.19 presents their joint collection.

Faced with this figure, the following things become apparent:

- The interlinking of content related overviews. This addresses the temporal coherence and unity as it was required by Def. 3. So, for example, it becomes clear by which scenario a certain network process will be manifested and finally becomes real in the iterative interplay of concrescence.
- The interplay of different kinds of systems. For example, it becomes clear how process systems and behavior-constructing systems are considered jointly, so that a certain network process is established (see *BPCOverview*).
- The hierarchical composition of the NEM. The joint neuronal knowledge process model (Def. 14) is broken down to separate views. So, the complexity of all-embracing visualizations is reduced, although the ANN systems operate on their usual level. This was defined in section 4.6.1.
- The codification of knowledge. As different kinds and forms of knowledge objects are broken down to the atomic level of knowledge, the codification for the knowledge processing units become apparent. By this, an interpretative knowledge use can be specified for a certain learning task, which corresponds to the foundations presented in section 2.5.3.

Aiming for the technical substantiation of process modeling entities by ANN techniques, so that a NPM, NPS, and NPO (see section 3.1.1) can be realized, with

**Figure 4.19** The Complex NMDL Taxonomy Views

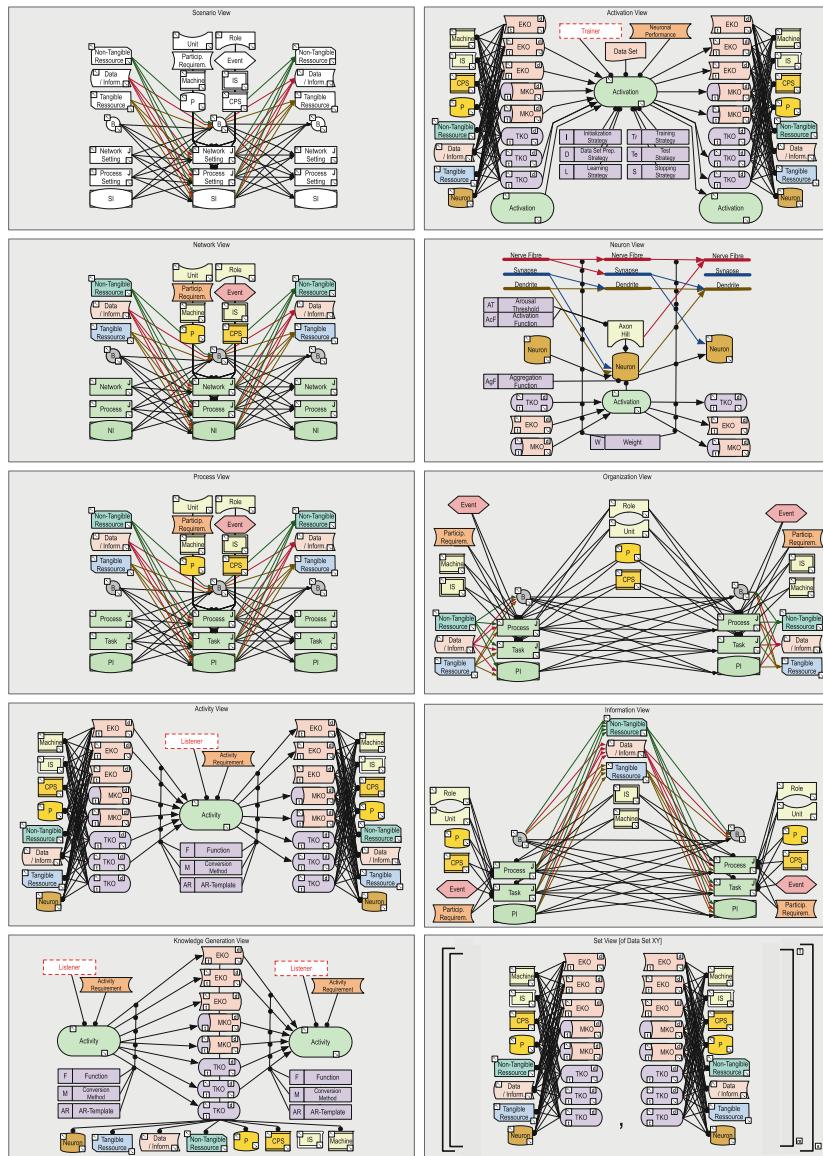
the aid of complex taxonomic views of the NMDL, any kinds of modeling objects and their mapping on the neuronal level becomes transparent. Hence, the NMDL issues modeling-relevant entities and makes them accessible for the process of ANN substantiation.

Flow-Oriented Views: The different flow-oriented views on the knowledge process model establish the isolated dynamic view, which issues a specific aspect. Fig. 4.20 presents their joint collection.

Faced with this figure, the following things become apparent:

- The learning task or rather training of an ANN (section 2.5.2) can be operationalized since relevant modeling items and corresponding attributes are provided by the *ActivationView*, e.g. This for example issues the choice of training mechanisms, data sets and stopping strategies. With this, the training can be specified by NMDL-based models.
- Since the training output is operationalized by the NMDL, the ANN's operation becomes accessible for the use as part of the NKMM (section 4.5). This, for example, includes the design of training settings by the *ActivationView*, the data collection from any kind of environmental setting by the *ScenarioView*, their interlinking by the *NetworkView* and its joint processual characterization by the *ProcessView*, as well as the ANN result use within any kind of symbiotic simulation setting by the *ActivityView* and the ANN-based processing by the *ActivationView*. All of them serve to function as metaphorical lever for management activities presented by the NKMM.
- Coming from the *NeuronView*, biologic mechanisms are defined according to section 2.5.1, so that these can be mapped to interlinked views. For the first, this enables the transformation of realistic models of third party simulators in the CoNM context with the NMDL. For the second, this provides the transformation of biological models to the ANN level of the pragmatic knowledge flow. For the third, this integrates neuronal mechanisms in the process context established by the NMDL's *ActivationView*, *ActivityView*, *ProcessView*, etc.
- Further, different kinds of neuronal architectures can be designed according to section 2.5.5 and section 2.5.8. Since the modeling items are substantiated by neuronal techniques, traditional as well as non-convenient architectures can be constructed on behalf of the NMDL in a drag-and-drop manner.

Grace to the interlinking of different flow-oriented views, various kinds of systems are integrated. Examples can be found in the following: (1) Neuronal simulators can be specified on behalf of the *NeuronView*. (2) Input by simulators and real-time

**Figure 4.20** The Flow-Oriented NMDL Views

systems can be specified by the *ScenarioView*, (3) ANN structures can be faced with different configuration settings according to the *NetworkView* and *ProcessView*. (4) The ANN system is characterized as knowledge carrier by the *ActivityView*. (5) The ANN system is specified by the *ActivationView*. (6) The ANN system and the corresponding knowledge generation is issued by the *KnowledgeGenerationView*. Since all of them are connected by the all-embracing NEM, they affect models and outcomes of different kinds of systems.

Grace to separation of perspectives and views on the NEM, different kinds of experts are enabled to cooperatively work on the same knowledge process model (Def. 14). Examples can be found in the following: (1) The Neuro-Science-Expert might focus on biologic mechanisms of the *NeuronView*. (2) The ANN expert might focus on the training and testing of ANN structures on behalf of the *ProcessANNView*. (3) The KM expert might focus on the knowledge and competence identification by the *KnowledgeGenerationView*. (4) The BP expert might focus on the process model designed (*ProcessView*). (5) The BP expert might focus on the process performance and controlling (*ProcessView*). (6) The management expert might focus on BP outputs on behalf of the *NetworkView*. (7) The simulation expert might focus on the scenario design with the aid of the *ScenarioView*. Since each considers individual views and proper perspectives on the joint neuronal enterprise model (section 4.6), the experts are allowed to focus on their expertise.

Grace to the focusing of ANN-based entities, e.g. by the *OrganizationView* and the *InformationView*, the systematic collection and storage of relevant knowledge is enabled. This corresponds to the knowledge base extension of an organization by neuronal interpretations of knowledge. As it becomes clear by which scenario and under which process conditions knowledge is generated, the target-oriented reuse of knowledge in different scenarios is enabled. This includes its deletion and creation within a set of processes according to the NKMM (section 4.5). This even supports the target-oriented ANN system creation, as attractive ANN structures can be identified and reused for implementations.

4.6.5 Interim Conclusion

The NMDL here designed intends to stand as a tool for the modeling and management of ANN systems. By the isolation of complex system aspects with the aid of perspectives and views, first, different types of experts are enabled to model cooperatively and focus on their familiar modeling expertise, second, different aspects of models become modifiable, so that experts are enabled to construct, reuse, and adapt models to a certain context, third, the black-box behavior of ANN systems

is brightened with the aid of a huge collection of modeling items with whom the inner working of ANN systems can be visualized. This includes, for the fourth, the knowledge generation by ANN systems and enables their consideration as *integration constant attributes*, which shall be transferred to the reality in the sense of an extension mapping (see Fig. 2.1).

As models visualized are substantiated by a corresponding ANN system, the following implications can be identified: First, ANN models become constructable in a drag-and-drop manner, since ANN systems reflect structures of traditional BP modeling mechanisms. Second, the relevant items incl. their sub-structures become accessible as well as modifiable, since modeling items reflect the current state of its corresponding ANN system. Third, different kinds of mechanisms become extendable by different types of experts, since each view focuses on individual technical substantiations. Forth, models about ANN systems can be constructed by the derivation from its corresponding systems. For example, the knowledge generated by a certain ANN system can be visualized by corresponding views, so that NPM (see Fig. 3.1) becomes transparent. Further, for the fifth, models of the ANN systems can visualize the activation of the corresponding ANN system, so that NPS (see Fig. 3.1) becomes transparent. Sixth, models allow to visualize the learning process of ANN systems, so that NPO (see Fig. 3.1) becomes transparent. This includes the NKMM-based measures (cf. section 4.5) as for example knowledge can be deleted by the deletion of associated modeling items and ANN system components.

Hence, the NMDL serves for the operationalization of ANN behavior. It can hence stand as communication platform for human-based and machine-based model constructors. It further functions as tool for the NKMM and makes the ANN operation observable and controllable within its current process context.

4.7 Systematic Exhaustion of Knowledge Objects

So that it becomes clear how the novel Concept of a Neuronal Modeling shall be used in order to systematically exhaust knowledge from ANNs, and further experiments demonstrating the CoNM use are transparent in regard with the knowledge extraction and this can be reproduced easily, the following addresses the objective of having a CoNM approach that guides the knowledge extraction from ANN by the CoNM (section 1.2). The algorithmic approach realizing a process modeling with the aid of ANN that focuses on the knowledge identification is referred to as Systematic Exhaustion of Knowledge Objects (SEKO). As various kinds of algorithmic approaches can be set up here, the following sections issue just one of

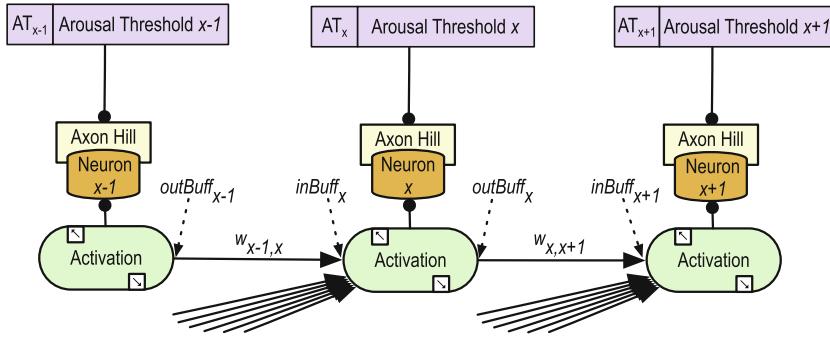


Figure 4.21 The Framework for Signal-Dependent Knowledge Detection (Neuron View)

many combinations, Fig. 4.21 presents a generalized framework for the technical implementation, on which the following attempts are oriented to.

On behalf of the NMDL shape specification presented at section 4.6.4, the figure focuses on the knowledge specification of neuron x . Hence, its preceding neurons $x - 1$ and its subsequent neurons $x + 1$ or rather their activations are presented, which are connected by knowledge flow edges having a weight w .

The history of a neuron is stored in buffer variables, so that each activation provides an *outBuff* for signals transmitted by a neuron or ANN via outgoing connections and an *inBuff* for the aggregated and weighted signals perceived by the neuron or ANN via all ingoing connections. Please remark here that because of the buffering of input and output values, signals coming from forward connections of time step t are treated aside signals coming from recursive connections of time step $t - 1$. Hence, a time step specific differentiation has not been visualized in Fig. 4.21 although it is technically supported and the figure subsumes both in the connection of neuron $x - 1$ and neuron x .

Since any neuron is associated with an axon hill, each neuron provides an arousal threshold $AT \in \mathbb{R}$. This represents a kind of activity that is relevant for the signal meaning for that specific neuron.

4.7.1 Static Knowledge Identification

The static knowledge identification (SKI) recognizes knowledge objects on behalf of the connection weights among neurons or rather the neuron's activations. As

this is not dependent on the current scenario the ANN is activated with, a signal-independent form of knowledge is identified, here. Hence, a knowledge object $+kb \in +KB$ is identified addressing the neuron's focus if a certain connection weight exceeds the arousal threshold in as much as the perceived signals are relevant (in regard with its incoming weights) and the transmitted signals are relevant (in regard with its outgoing weights). Further, a knowledge object $-kb \in -KB$ is identified addressing the neurons disregarding if a certain connection weight drops the arousal threshold in as much as the perceived signals are relevant to be disregarded (in regard with its incoming weights) and the transmitted signals are relevant to be disregarded (in regard with its outgoing weights). So, as part of the conversion $Conv$, we can identify the input objects $conv_{in} \forall w_{x-1,x}$

$$conv_{in} = \begin{cases} +kb, & \text{if } w_{x-1,x} \geq AT_x \\ -kb, & \text{if } w_{x-1,x} < AT_x \end{cases} \quad (4.98)$$

and we can identify the output objects $conv_{out} \forall w_{x,x+1}$

$$conv_{out} = \begin{cases} +kb, & \text{if } w_{x,x+1} \geq AT_x \\ -kb, & \text{if } w_{x,x+1} < AT_x. \end{cases} \quad (4.99)$$

Variations of a static knowledge identification can consider the arousal threshold of the consecutive neuron for the knowledge identification of $conv_{out}$ at neuron x , or they can consider the arousal threshold of the preceding neuron for the knowledge identification of $conv_{in}$ at neuron x . Regarding the positiveness (exhibition) or negativeness (inhibition) of a connection's weight, based on a static knowledge identification, one further can derive in how far the sign of a signal is reversed or not. As this is codified by the color of the corresponding knowledge flow edges, this kind of knowledge will not be objectified as individual knowledge objects in a first CoNM version.

4.7.2 Dynamic Knowledge Identification

The dynamic knowledge identification (DKI) recognizes knowledge objects on behalf of the signaling among neurons or rather the neuron's activations and follows (Gronau and Grum, 2017; Grum and Gronau, 2018a). As this is dependent on the current scenario the ANN is activated with, a signal-dependent form of knowledge is identified, here. Hence, a knowledge object $+kb \in +KB$ is identified address-

ing the neuron's focus if a certain activation signal exceeds the arousal threshold in as much as the perceived signals are relevant (in regard with its preceding neuron output buffer) and the transmitted signals are relevant (in regard with its own output buffer). Further, a knowledge object $-kb \in -KB$ is identified addressing the neurons disregarding if a certain connection weight drops the arousal threshold in as much as the perceived signals are relevant to be disregarded (in regard with its preceding neuron output buffer) and the transmitted signals are relevant to be disregarded (in regard with its own output buffer). So, for each moment m , as part of the conversion $Conv$, we can identify the input objects $conv_{in} \forall m$

$$Conv_{in} = \begin{cases} +KB, & \text{if } outBuff_{x-1} \geq AT_x \\ -KB, & \text{if } outBuff_{x-1} < AT_x \end{cases} \quad (4.100)$$

and we can identify the output objects $conv_{out} \forall m$

$$Conv_{out} = \begin{cases} +KB, & \text{if } outBuff_x \geq AT_x \\ -KB, & \text{if } outBuff_x < AT_x. \end{cases} \quad (4.101)$$

Variations of a dynamic knowledge identification can consider the arousal threshold of the consecutive neuron for the knowledge identification of $conv_{out}$ at neuron x , or they can consider the arousal threshold of the preceding neuron for the knowledge identification of $conv_{in}$ at neuron x . Further, one can consider the neuron's input buffers, so that the joint interplay of preceding neurons is issued. Furthermore the use of signal weight combinations can be addressed, so that the structural and dynamic kinds of knowledge are considered jointly. Of course, arousal thresholds can be more sophisticated, such as manifest as behavior as well, and the effect of exceeding a threshold can be regarded in regard with the consequence of the total ANN performance. For a first CoNM version, this shall refer to a simple number addressing neuron x only.

4.7.3 Interim Conclusion

In order to evaluate the SKI meaning of SEKO in the CoNM context, the following relations can be identified: First, in the context of NPM, the SKI extracts knowledge codified in the ANN structures. Faced with training runs changing the structural consistence of ANN, by this, the current knowledge contained in ANN becomes transparent. In the context of NPS, the SKI indicates which kinds of simulation

activation is relevant at a certain neuron or ANN. In the context of NPO, the SKI clarifies by the direct comparison of knowledge identified before and after an optimization attempt, what structural changes are responsible for an improvement or worsening.

In order to evaluate the DKI meaning of SEKO in the CoNM context, the following relations can be identified: First, in the context of NPM, the DKI extracts knowledge codified in the ANN behavior, when it is faced with a scenario. Faced with training runs changing the structural consistence of ANN, by this, the effect of the current ANN structures becomes transparent. In the context of NPS, the DKI indicates which concrete simulation activation is relevant at a certain neuron or ANN. In the context of NPO, the DKI clarifies by the direct comparison of knowledge pattern identified before and after an optimization attempt, what behavioral changes are responsible for an improvement or worsening.

As by the DKI the objectification of activation-based signals is realized, knowledge objects refer to patterns of signals causing a certain behavior. This holds a potential for the symbiotic KM (section 4.5), as the organizational knowledge base is enriched by neuronal knowledge (1), a training can be based on knowledge objects generated (2), training can focus on ANN compartments (3) and an interactive controlling by human knowledge managers and process managers is enabled (4).

Faced with the CoNM meta-model presented in Fig. 4.4, a DKI can complete NMDL models for individual moments, for the joint sequence, for all sequences and even for a collection of data sets. Further, although $-KB$ is essential for the joint ANN performance, one might be particularly be interested in $+KB$, so that a hiding of the first kind of knowledge supports smaller models. A CoNM implementation therefore needs to provide corresponding filtering options.

4.8 Learning Principles in CoNM

In order to consider the biological plausible learning potential of ANN in the framework of the CoNM, the following structures relevant new designs. It considers the hierarchical composition of systems as it was designed in section 4.6 and reflects neuronal process circles of Fig. 4.10. Therefore, it becomes clear how the novel Concept of a Neuronal Modeling shall be used as learning principle and further experiments demonstrating the CoNM use are transparent in regard with their training procedures and can be reproduced easily, the following addresses the objective of having a CoNM as learning principle that guides the CoNM realization in ANN training, modeling, simulation and process optimization (cf. research objectives of section 1.2).

4.8.1 Principle of Neuronal Recursion

Recognizing the approximation potential of ANN and having the hierarchical framework of the NMDL, the abstract proceeding of applying ANN constructions on an initial ANN until an abort criterion is reached is referred to as *Learning Principle of Neuronal Recursion* (LPNR). An example can be found at the adding of further neurons on more atomic levels, so that the desired learning capacity and approximation performance is reached.

In the sense of the cooperative modeling (cf. section 4.5), this can be realized by both, human as well as machine model creators. So, while setting up BP by process experts, their mapping via process networks, processes, tasks, activities and activations to an arbitrary complex network of neurons can be realized by ANN of increasing size.

Since the recursion results in more neurons of a rising granularity, by a decreasing abstraction level, the complexity of models is decreased without worsening the ANN performance because of clean models.

4.8.2 ANN-based Simulation Systems

The CoNM considers ANN as knowledge carriers within their processual context of business processes. In a first version, it considers simulations to be a time-oriented method (cf. section 2.3.2.10), which means the following:

Simulation time and ANN activation are in sync.

Hence, one simulation step corresponds to the activation of corresponding ANN structures. Being faced with live simulations, ideally, the data provision of live systems is in sync with the simulation system. If this is not the case, the following two variants can be found:

- 1) Either a live system provides more data more frequently than the simulation system requires. Here, only that data is gathered from live systems that currently is available. From a pedagogic design, this means the loss of valuable information occurring meanwhile data measurement points.
- 2) Alternatively, the live system provides less data than the simulation system requires. Then, the simulation system considers the last current data gathered from the live system.

Based on the understanding of ANN to function as actual entity (Def. 3) and to form in a multiplicity the actual existing world, their interplay of concrescence in

the private is interpreted to be the simulation system operation and their interplay of concrescence in the public is interpreted to be the simulation system output. On behalf of contemporary mathematical models of ANN systems, the imitation of the real-world processes can be imitated by a neuronal simulation system. So, in accordance with the simulation definition set up (Def. 16), these kinds of simulations are referred to as *neuronal process simulation*. In the context of these kinds of simulation systems, a learning focuses on the improvement of interconnected ANN systems.

4.8.3 NPO Backpropagation

Based on the understanding of ANN to function as approximation of an organization, on behalf of mathematical learning principles for the training of ANN systems, a learning organization in the sense of section 2.4.1.2 is set up, that is in accordance with Def. 19. Errors are detected and corrected by the *neuronal process optimization*. By this, the neuronal knowledge base is updated, extracted by SEKO approaches (cf. section 4.7) and has been demonstrated by the experiments of section 6.4.

In the sense of a symbiotic KM (cf. section 4.5), on base of NPO modifications, interventions for human knowledge carriers can be derived and their competences can be enriched. In accordance with the optimization philosophy of CIP set up (section 2.4.2.2), these kinds of optimizations are a neuronal form of CIP if they are realized continuously (compare with Fig. 4.10). However, based on this understanding, the use of algorithmic training procedures for ANN are plausible.

Beside algorithmic training variants of ANN presented in section 2.5.6.1, a back-propagation variant was designed, that respects trained compartments of individual ANN systems and focuses on the training of more complex systems. Labeled with *NPO Backpropagation*, it is based on weight adjustments by Eq. 2.44 as well as its time-sensitive variant and requires the following steps:

1. Input patterns are fed into the ANN and propagated through all layers.
2. The network activation is compared with the intended activation, so that an error is derived.
3. The error of the output layer is propagated from the very last layer to the very first layer, so that weight adjustments can be derived for each layer.
4. Weight adjustments are carried out only for selected weights, which depends on the operational NPO level. As such, this can refer to super systems, such as more complex activations, activities, tasks, processes and networks, that interconnect

pre-trained ANN. Further, this can refer to the modification of weights being within a certain area of space.

Faced with the NPOC (cf. Fig. 4.10), the *NPO Backpropagation* supports the identification of irrelevant process compartments, so that they can be removed and NPS improve without changing more atomic knowledge bearers.

Alternatively, one can use standard training procedures for optimizing neuronal process models, too. Since these won't respect pre-trained ANN being interconnected, the joint ANN will change. On the other hand, this can result in more precise and robust ANN, because all the connection are allowed to change. Finally, a trade-off between the crystallization of knowledge by the use of *NPO Backpropagation* and the consideration of fluent knowledge by the use of standard *Backpropagation* procedures must be found.

4.8.4 Interim Conclusion

In order to evaluate the meaning of learning principles presented in the CoNM context, the following relations can be identified:

First, the learning principles enable the NPS because sub-systems can be interconnected and function as joint ANN simulation system. Second, individual ANN can function as isolated simulation objects. Third, learning principles enable the NPO because weight modifications can be focused on the relevant edges interconnecting more complex super systems. Fourth, sub-systems can be approximated recursively on an arbitrary level of detail.



Implementation

5

While the design of the previous section intends to conceptually consider the objectives of this contribution, the implementation partly realizes the concepts defined and implicitly considers the objectives of this contribution. For their complete realization, the implemented tools need to be demonstrated.

As the research carried out intends to follow the DSRM of Peffers (details can be found in section 3.3), the implementation can be seen as a form of demonstration too. Hence, the artifacts implemented will be jointly evaluated with their demonstration projects by the corresponding CoNM-SESS (section 7.1.3). With the intention of introducing tools to carry out the CoNM, this section is structured as follows:

First, the usability concepts are presented, which have been implemented by corresponding prototypes. Second, the architectural set up is presented, which has been implemented and characterizes the technical environment of the corresponding prototypes. Finally, the software prototypes are presented using which the CoNM can be carried out. Corresponding demonstrations will be presented in section 6. Since the demonstrations shall focus on the functionality of the CoNM, tool-based issues have been presented in this section beforehand.

5.1 Usability Concept Realization

The CoNM intends to apply the tempo-spatial relations of biological systems such as the human brain to the context of business processes; for instance, the CoNM integrates different kinds of modeling worlds. Hence, in accordance with the

Supplementary Information The online version contains supplementary material, such as Appendix A, B, C and D whose references are indicated in the text, available at https://doi.org/10.1007/978-3-658-35999-7_5.

requirements, a new kind of CoNM software bunch needs to be complex and designed to enable comfortable access to the CoNM elements from a specific modeling world and individual perspective.

As a guidance for the development of hardware and software systems, first, a new kind of modeling world integration has been presented, which is called *bidirectional modeling in XR*. Then, relevant modeling use cases have been collected to reflect them through corresponding hardware and software systems.

5.1.1 Bidirectional Modeling in XR

While the traditional process modeling in 2D was already captured quite a long time ago, which was described in section 2.1, modeling within AR started evolving only recently.

The first conceptional ideas on how to implement process modeling within the AR were presented by Grum and Gronau (2017), who prepared 3D augmentations of process models within the real world by using a ground plan and a sketch plan. For this, new process modeling tool requirements were collected so that corresponding modeling tool extensions could be derived successfully. These served the prototype implementation that followed.

In order to overcome the modeling limitations regarding the interplay of the common process modeling world in 2D and the augmented world in 3D systematically, the concept of a *bidirectional modeling* was designed by them. This enables the exchange of modeling activities from a 2D modeling world to a 3D AR modeling world and vice versa without any hindrance. It not only represents the most complex modeling mode but also the most interactive modeling scenario. While detailed definitions of the four modeling modes, six modeling scenarios and five dimensions for their characterization can be found in the corresponding conference paper (Grum and Gronau, 2018b), an overview has been presented in Fig. 5.1.

As one can see in this figure, models can be constructed using four modeling modes, which have been derived from the modeling scenarios and dimensions as described below.

In the first mode, models are constructed in a *no-interplay-mode*, which allows modeling only within the 2D modeling world or within the 3D AR modeling world by one person. Afterward, without the need of having synchronous interactions, the models can be visualized by other persons from the same modeling world. While a majority of the BPM tools can be classified in this category, modeling within the 3D AR modeling world using this mode can be identified as an innovation.

Alternatively, in the second, models are constructed in a *one-person-mode*. Here, the model constructor recognizes a modeling in one of the two modeling worlds.

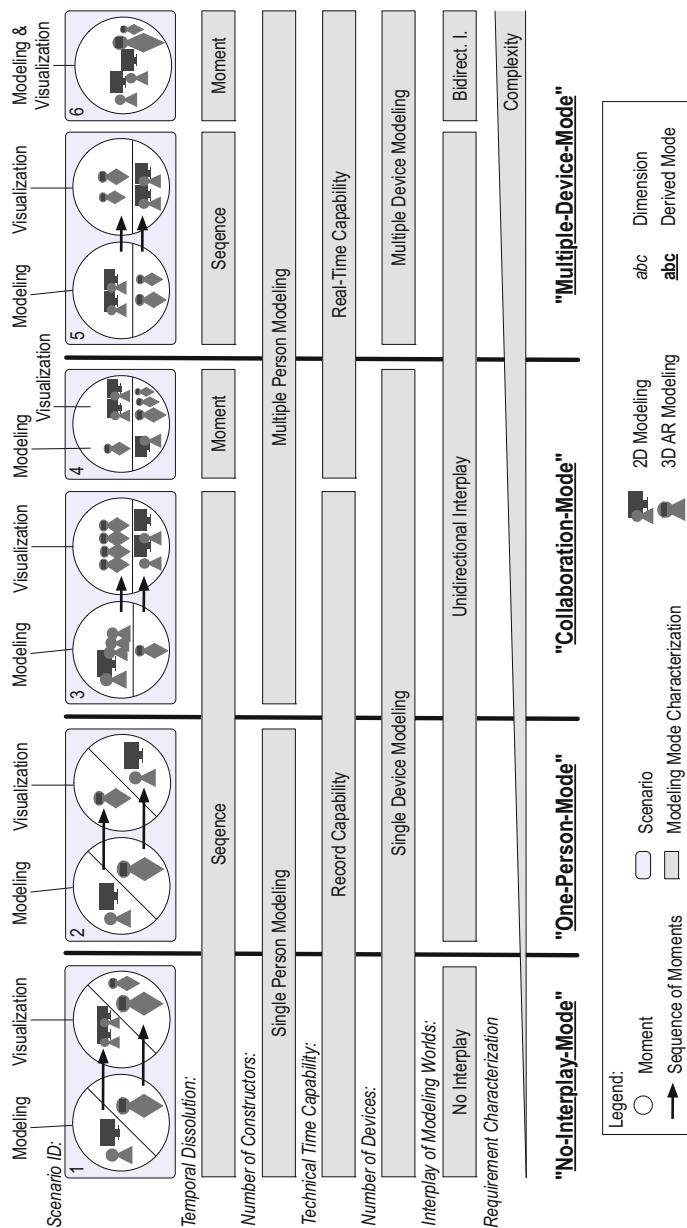


Figure 5.1 Common Modeling Modes for the Bidirectional Modeling (following Grum and Gronau, 2018b)

Afterward, the visualization is enabled either in the same or in the other modeling world.

Using the third mode, models are constructed in a *collaboration-mode* that realizes a modeling in both modeling worlds, where only one person carries out modeling and others give feedback based on visualizations. Please note that in a more sophisticated version (see the fourth scenario), the modeling of a constructor and the visualization of the current model by others is carried out at the same time.

In the fourth modeling mode, the *multiple-device-mode*, various model creators construct models cooperatively using different kinds of modeling devices. In the most sophisticated scenario of this mode (see the sixth scenario), modeling and visualization are carried out simultaneously, which is the most effective and limitation-free bidirectional modeling. Please note that only innovative BPM tools support simultaneous cooperative modeling within the 2D world, which is not issued in this category, as the focus is set on the integration of all the modeling worlds. Currently, none of the BPM tools consider the interplay with the AR world issued here.

Since the complexity of implementations and the number of requirements for an implementation depend on the characterization of the dimensions identified, as reflected in the corresponding scenario number as well, a step-wise implementation is recommended (Grum and Gronau, 2018b, p. 109). Hence, this recommendation has been used as a guidance for designing the prototype presented in the following sections.

5.1.2 XR Modeling Use Cases

Faced with the potential of the CoNM to consider tempo-spatial relations in 2D and 3D within virtual reality (VR) and AR (section 4.6.2 considers the points O, P, Q, R), the following addresses the modeling in any kind of reality (XR). Based on Def. 5 (modeling), this includes engineering activities that improve model modification in a repetitive and systematic manner because they all affect the final model constructed. Hence, the following includes activities on model simulation and optimization.

The collection of relevant use cases builds on the need to have two kinds of devices:

1. Devices addressing virtual reality content, such as those provided by traditional process modeling; for example, desktops, laptops and smartphones.
2. Devices addressing augmentations, such as the positioning of modeling items within the real world; for example, AR glasses, smartphones, tablets and laptops.

Fig. 5.2 presents the most essential use cases for both kinds of devices, thus depicting the actor who is faced by the two different kinds of devices as well as by various kinds of tools, corresponding model types and modeling languages.

Following a variant of UML's use case diagram type, in the figure, one can see the human actor as the central element. It must be considered in one of its roles of the NKMM (see section 4.5.5), which refers to *manager*, *knowledge manager*, *knowledge worker*, *process ANN manager* and *process ANN worker* and includes different kinds of experts, such as biologists, simulation experts, ANN experts and KM experts. Corresponding to the actor's role and expertise, it uses the first kind of VR devices in order to carry out VR-based modeling activities. This considers all the kinds of models being issued by the CoNM and includes the NKMM activities presented in section 4.5.1. Further, the actor uses devices that address augmentations in order to carry out the same kinds of activities within AR. As a UML variant, these have been visualized by the well-known artifact shape.

When dealing with these two kinds of tools, the difference in their focus is clear; while traditional VR tools support traditional modeling activities and focus on the 2D modeling world, they provide a great level of detail. Examples of traditional tools include BPM tools, KM tools, Group and Workflow tools, simulation tools and, now, CoNM relevant tools. In contrast, augmentation tools focus on the 3D AR world but do not provide a good level of detail. Their primary purpose is to bidirectionally transport information from and to the real world. Because of the cumbersome text input and the lack of operative control mechanisms as of now, they mainly limit themselves to the controlling of sophisticated functions of VR-based tools by shortcuts. This excludes functions that are not required to be positioned within the real world. Hence, as the first CoNM version, the latter kind of tools will complement the former. Next, CoNM versions will have to examine how far AR-based tools are capable of replacing the first kind of tools or of operating as a stand-alone tool.

Addressing the CoNM terminology presented in section 3.1.1, and assuming it has background servers carrying CoNM-relevant concepts and techniques, first, the NPM is provided by the case *realize modeling from BPM tool* as well as the case *realize modeling from Augmentor* since machine-based models are viewed by the tools and a modification of the corresponding models is enabled. Second, the NPS is provided by the case *realize VR simulation from BPM tool* as well as the case *realize AR simulation in Augmentor*. Third, the NPO is provided by the case *realize optimization from BPM tool* as well as the case *realize optimization in Augmentor*.

When looking at the architectural neuronal process management fields of the NEA introduced in section 4.3, one can identify use cases addressing the first three layers of both kinds of devices. An association has been indicated by the roman

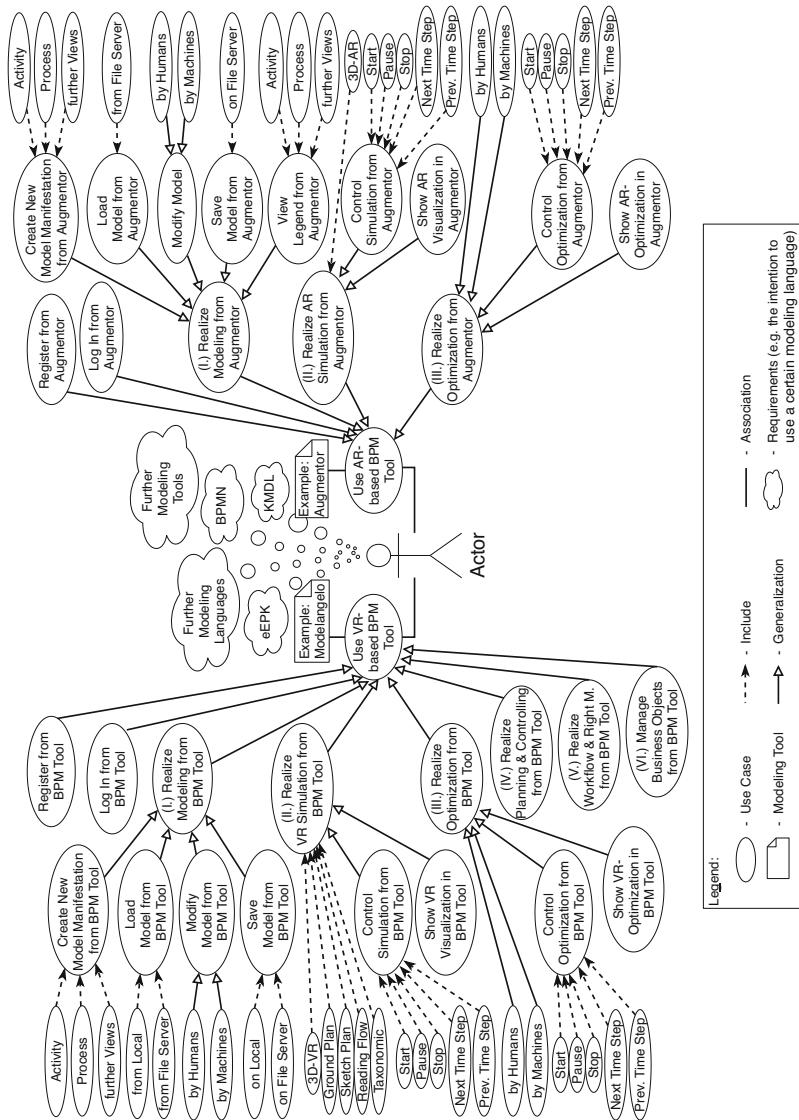


Figure 5.2 Common Use Cases for a Bidirectional AR Modeling (following Grum and Gronau, 2018b)

numbering of corresponding use cases in Fig. 5.2. As the NEA levels IV–VI require functions that are more sophisticated than the simple shortcuts that could be realized at the Augmentor, the use cases addressing these kind of NEA levels are only addressed by VR devices, such as by workstations, laptops or desktops.

5.1.3 Interim Conclusion

The bidirectional XR modeling was conceptualized based on how different kinds of modeling worlds can be integrated. Guided by an ordered list of extension levels, as shown in Fig. 5.1, the most attractive but complex level of the *multiple-device mode* enables the most flexible interactions. Therefore, it holds the greatest potential for the CoNM. Faced with numerous interim levels, the CoNM prototypes will have to focus on the least complex level that is able to clarify the CoNM functioning.

Faced with the XR modeling use cases, the kind of relevant modeling scenarios that need to be reflected by the CoNM software and hardware were identified. When dealing with a great number of cases, the CoNM prototypes will have to focus on the most relevant cases. These do not include administration processes, as these are trivial. Therefore, the CoNM prototypes focus on NPM-related, NPS-related and NPO-related use cases that clarify the CoNM functioning.

Considering the modeling modes discussed in section 5.1.1 and the use cases of section 5.1.2, the usability concept for the CoNM tools has been established. They will serve as a guidance for the development of corresponding hardware and software systems as presented below.

5.2 Architectural Set Up

To set up a technical environment capable of carrying out the CoNM and to demonstrate the use of CoNM software prototypes, this section instantiates and implements the NEA defined in section 4.3. While Fig. 4.8 depicts the architectural design of the NEA and differentiates components based on content, the following focuses on the concrete technical components, such as its servers, devices, hardware and software components, as well as its corresponding infrastructure. Hence, it becomes clear how architectural fields can be mapped on the IT infrastructure.

So, following the NEA defined, Fig. 5.3 presents one of many possible technical instances and clarifies the architectural set up for the prototypes implemented. Here, the naming of the elements visualized is based on the naming presented in Fig. 4.8.

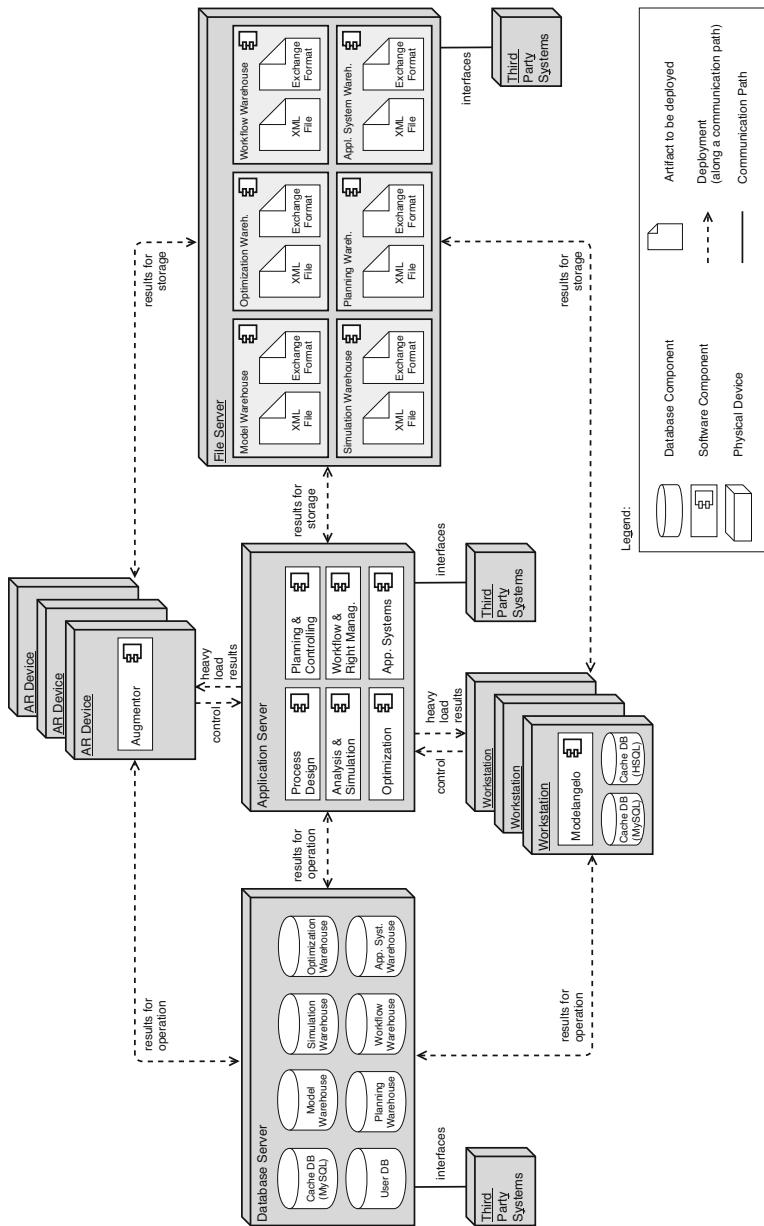


Figure 5.3 Architectural Set Up (as UML Deployment Diagram Variant)

Following a variant of the UML deployment diagram type, in the figure, one can see the physical deployment of artifacts on nodes. By this, it becomes clear which software elements are provided by a certain hardware element. The hardware components, “nodes”, have been visualized by UML’s 3D cubes and refer to web servers, application servers and database servers; for instance, software components and “artifacts” run on those nodes. The latter have been visualized by UML’s component shapes. As a variant, database components have been visualized by a well-known database symbol; furthermore, artifacts being deployed have been visualized by well-known document items. The following presents hardware nodes by individual sub-sections.

5.2.1 Devices

Fig. 5.3 clarifies that various AR modeling devices carrying the software prototype *Augmentor* and various VR modeling devices carrying the software prototype *Modelangelo* enable the CoNM access; they control the heavy operations carried out by the application server and visualize corresponding processing results.

5.2.2 Application Servers

Regarding the NEA levels presented at Fig. 4.8, the architectural set up maps any architectural neuronal process management field, including its framework components on an individual processing system (application), all being provided by an application server. As ANN training demands different hardware characteristics from the ANN inferences or corresponding simulations, each software component principally demands a specialized server. As the prototyping does not issue performance but technical functioning, the optimal hardware dimension has not been addressed any further.

5.2.3 Database Servers

As various devices are intended to operate on the same model simultaneously (demanded by the complex modes of bidirectional modeling, cf. section 5.1.1), current processing results are cached by individual databases. These are either deployed from and to the collection of devices, or they are deployed from and to the applications carried by the application server. Hence, for any NEA field-specific component

warehouse presented in Fig. 4.8, an individual database has been issued. So, Fig. 5.3 presents separate database components for them.

5.2.4 File Servers

Since different devices are intended to store various kinds of models systematically, (described by the use cases of section 5.1.2), current processing results are stored by individual file systems. These are either deployed from and to the collection of devices, or they are deployed from and to the applications carried by the application server. Hence, for any NEA field-specific component warehouse presented in Fig. 4.8, an individual file system has been issued. So, Fig. 5.3 presents separate components for them.

5.2.5 Third Party Systems

Considering the various kinds of NEA interfaces, in accordance with Fig. 4.8, these are referred to as field-specific framework component interfaces, wherein corresponding third party systems can be integrated using specialized communication paths. By this, a communication from the CoNM system to external systems can be realized, such as the simulation result use in external simulation systems, workflow impulses instructing external workflow systems or prediction results being considered in external planning systems. Further, communication from external systems to the CoNM system can be realized. Some examples of this include a CoNM business object update because of a change in IT architectural systems, the simulation input in the sense of a symbiotic simulation, or the parameter stream from production systems and its use in NPS. By this, the NEM is enabled to function as a neuronal control center affecting real-world objects.

5.2.6 Interim Conclusion

The architectural set up designed here intends to carry software prototypes presented in the following section. Based on this architecture, the implementation requirements have been fulfilled as shown in the following. First, the cooperative model construction demanded by the bidirectional XR modeling is shown (section 5.1.1). Second, according to the devices conceptualized and required by the CoNM modeling scenarios, the architecture carries relevant use cases and fulfills the requirements

that have been derived by CoNM use cases (section 5.1.2). Third, the requirements derived from the artifact of the NEA designed have been fulfilled, as the hardware implementation instantiates the NEA components presented in section 4.3. Fourth, the concept of the NEM is enabled (section 4.6) since third party systems are integrated via interfaces, and CoNM systems can function as organizational control center. They perceive environmental signals, and they affect organizational entities as actuators via workflow systems.

Together with being faced with the tool selections of the CoNM-SESS (section 6.1) as well as with its evaluations (section 7.1.1), attractive software components can be carried by the architectural set up implemented. Hence, tools such as the Modelangelo and PyBrain are ready for an extension.

5.3 Software Prototypes

Based on the qualitative analysis and tool selection presented in section 2.6 and considering the quantitative analysis and tool selection (presented in section 6.1), as well as their evaluation (presented in section 7.1.1), this section presents software prototypes that intend to carry the CoNM. More specifically, it exemplifies how to combine the tools called *Modelangelo*, *PyBrain*, the *Augmentor* and *Transformation* in order to enable the CoNM.

5.3.1 Augmentor Development

Regarding the intention to bring the CoNM within the context of the real world and the fact that process modeling was not issued in AR before, a software called *Augmentor* has been set up as a new development.

Following Zwicky's method of a general morphological analysis (Zwicky, 1966), a morphological box was constructed by Grum and Gronau (2018b), which parameterizes the complex problem of implementing this novel software. Fig. 5.4 presents an overview of the process to identify the best tool configuration to be implemented.

For the design of this box, the following sub-artifacts have been considered: 1) the meta-model, which resembles the ANN process taxonomy and the CoNM-meta-model (section 4.1.3); 2) the XR modeling modes for overcoming the modeling limitations of the common process modeling world in VR-2D and the innovative VR-3D and AR-3D modeling required by the CoNM (Fig. 5.1); and 3) relevant use cases of a CoNM-required modeling with AR devices and VR devices (Fig. 5.2). Following this, according to the methodology of Zwicky, its dimensions and manifestations

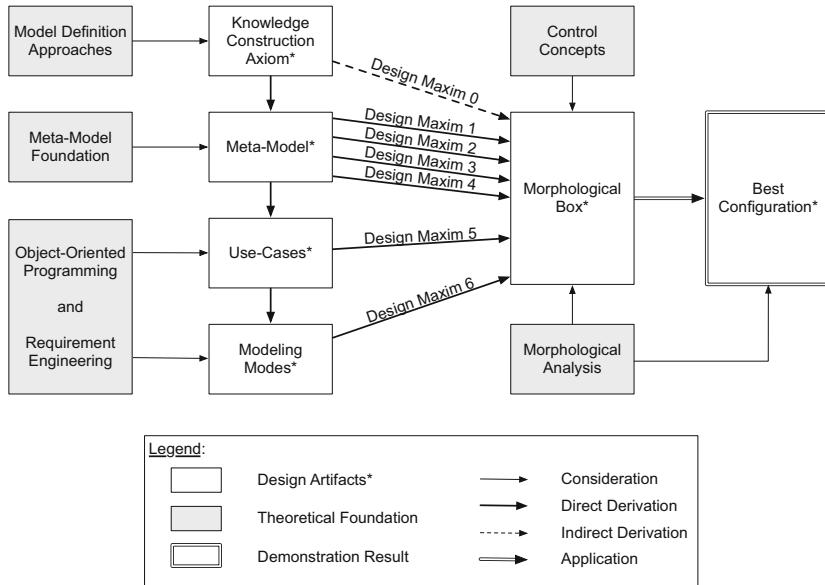


Figure 5.4 Morphological Box for a Bidirectional AR Modeling (following Grum and Gronau, 2018b)

per dimension were evaluated by 63 individuals educated in Business Process Management (BPM) at the University of Potsdam for one semester. This group included students with academic backgrounds in economics, computer science and business informatics, so that the best configuration could be identified. Fig. 5.5 presents the final morphological box and highlights the greatest acceptance of test persons in yellow.

In the figure, one can see the parameters and dimensions that were validated by experts; in total, 132 different configurations are shown. Based on the empirical evaluation and statistical analysis that determined the greatest acceptance on behalf of the majority, the Augmentor prototype has been set up.

Choice of Use Cases: Beside trivial use cases, such as the registration routine, the login process or the file loading and storing pattern, the following addresses the most relevant cases for clarification and functional proof-of-concept. Regarding

Origin	Dimension	Parameter				
Meta-Model	Modeling Object Focus	only time-based position related objects	perspective related objects	all objects		
Use Case Diagram for "Process Design" (NEA Level I)	Choose Intended Shape	per cursor and touch pad activation from legend	per physical look on AR modeling belt	per voice recognition	per thoughts and unique shape denomination	
	Select Intended Shape	per cursor and touch pad activation	per visual contact and eye focus analysis	per voice recognition and unique shape denomination	per thoughts and unique shape denomination	
	Grab Intended Shape	per touch pad activation	per image analysis and grab movement	per eye blink pattern	per voice recognition	per thoughts about grab instruction
	Position Intended Shape	per touch slide along axes	per movement within reality (sensory information, image analysis)	per voice recognition (axis and distance instructions)	per thoughts about position instruction	
	Drop Intended Shape	per touch pad activation	per image analysis and drop movement	per eye blink pattern	per voice recognition	per thoughts about drop instruction
	Delete Intended Shape	per shape selection and delete button activation	per image analysis and delete movement	per eye blink pattern	per voice recognition	per thoughts about delete intended shape instruction
	Connect Intended Shape	per cursor on connection visualization and touch pad activation	per relation object selection (see "choose intended shape")	per image analysis and connect movement	per eye blink pattern	per voice recognition
	Zoom On Intended Shape	per cursor and touch pad activation using two fingers	per image analysis and zoom movement	per eye blink pattern	per voice recognition	per thoughts about zoom instruction
	Modify Size of Intended Shape	per shape and axis selection and touch pad activation using two fingers	per image analysis and size modification movement	per eye blink pattern	per voice recognition	per thoughts about increase size instruction
	Rotate Intended Shape	per shape and axis selection and touch pad activation using three fingers	per image analysis and rotate movement	per eye blink pattern	per voice recognition	per thoughts about rotate instruction
	Undo	per button selection and touch pad activation	per image analysis and undo movement	per eye blink pattern	per voice recognition	per thoughts about undo instruction
	Redo	per button selection and touch pad activation	per image analysis and redo movement	per eye blink pattern	per voice recognition	per thoughts about redo instruction
Use Case Diagram for "Process Analysis and Simulation" (NEA Level II) and "Process Optimization, Training and Testing" (NEA Level III)	Start Simulation / Optimization	per cursor on start button and touch pad activation	per image analysis and start movement	per eye focus on start button and blink pattern	per voice recognition of start instruction	per thoughts about start instruction
	Pause Simulation / Optimization	per cursor on pause button and touch pad activation	per image analysis and pause movement	per eye focus on pause button and blink pattern	per voice recognition of pause instruction	per thoughts about pause instruction
	Stop Simulation / Optimization	per cursor on stop button and touch pad activation	per image analysis and stop movement	per eye focus on stop button and blink pattern	per voice recognition of stop instruction	per thoughts about stop instruction
	Next Time Step / Iteration of Simulation / Optimization	per cursor on next button and touch pad activation	per image analysis and next movement	per eye focus on next button and blink pattern	per voice recognition of next instruction	per thoughts about next instruction
	Previous Time Step / Iteration of Simulation / Optimization	per cursor on previous button and touch pad activation	per image analysis and previous movement	per eye focus on previous button and blink pattern	per voice recognition of previous instruction	per thoughts about previous instruction
Use Case Diagram for "Administration"	Register	per digital keyboard and key selection (see "choose intended shape") in guided register process	per movement on key of digital keyboard and image analysis in guided register process	per eye focus on key of digital keyboard and blink pattern in guided register process	per voice recognition in guided register process	per thoughts in guided register process
	Login	per digital keyboard and key selection (see "choose intended shape") in guided login process	per movement on key of digital keyboard and image analysis in guided login process	per eye focus on key of digital keyboard and blink pattern in guided login process	per voice recognition in guided login process	per thoughts in guided login process
	Create New Model	per digital keyboard and key selection (see "choose intended shape") in guided create process	per movement on key of digital keyboard and image analysis in guided create process	per eye focus on key of digital keyboard and blink pattern in guided create process	per voice recognition in guided create process	per thoughts in guided create process
	Load Model	per digital keyboard and key selection (see "choose intended shape") in guided load process	per movement on key of digital keyboard and image analysis in guided load process	per eye focus on key of digital keyboard and blink pattern in guided load process	per voice recognition in guided load process	per thoughts in guided load process
	Save Model	per save button selection (see "choose intended shape")	per movement on key of digital keyboard and image analysis in guided save process	per eye focus on save button and blink pattern in guided register process	per voice recognition in guided register process	per thoughts about save instruction
Modeling Modes	Modeling Scenario Id	1	2	3	4	5
	Temporal Dissolution of Modeling Activities	Moments	Sequences			6
	Number of Devices Modifying Models	Single Person Modeling	Multiple Person Modeling			
	Technical Time Capabilities	Record Capability	Real-Time Capability			
	Number of Devices Used for Modifying Models	Single Device Modeling	Multiple Device Modeling			
	Modeling World Interplay	No Interplay	Unidirectional Interplay	Bidirectional Interplay		
	AR Modeling Modes	Unidirectional-Interplay-Mode	One-Person-Mode	Collaboration Mode	Multiple-Device-Mode	

Figure 5.5 Morphological Box for a Bidirectional AR Modeling (following Grum and Gronau, 2018b)

the NPM, NPS and NPO, these refer to the first three levels of the NEA designed (section 4.3).

Use Case—Realize Modeling form Augmentor:

According to the first level of the NEA designed (*Process Design*, as shown in Fig. 4.8), the Augmentor visualizes the anchoring of process modeling items within the real world. Fig. 5.6 presents an impression of the software implemented.

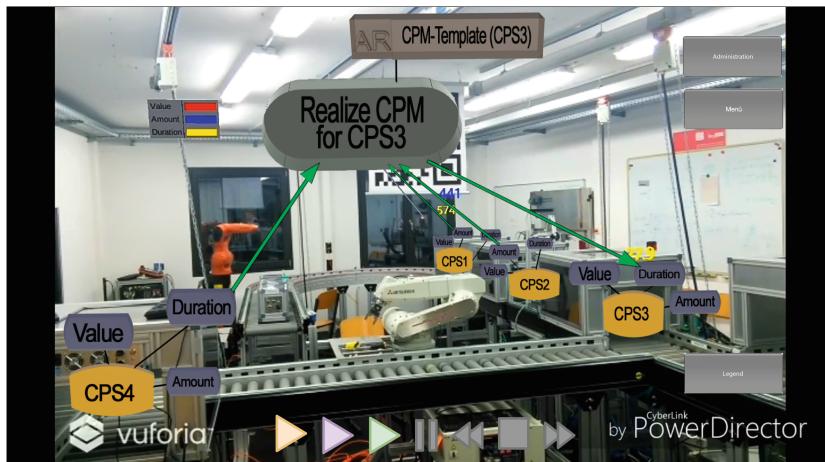


Figure 5.6 AR Modeling on behalf of the Augmentor (Example from Grum and Gronau, 2017)

As the Augmentor's focus lies on information gathering within the real world, information for the points O , P , Q and R (defined in section 4.6.1 and section 4.6.2) can be collected efficiently for any kind of modeling object. The Augmentor's focus also lies on the visualization of the 3D shapes of an object and their animation. So far, these are anchored based on the identification of AR target images by the Vuforia framework. The modeling objects simply can be chosen from the legend of the corresponding view and dropped within the real world. By this, the use of VR-based modeling tools is complemented. Fig. 5.6 shows the 3D-AR visualization of an Industry 4.0 knowledge generation. The corresponding ground plan, sketch plan, reading-flow-optimized model and two more examples can be found at (Grum and Gronau, 2017).

Further, the Augmentor issues a second modeling use case variant, which refers to the modeling by machine-based model creators (modeling Def. 5), meaning the ANN systems. This is accessed by the activation of the corresponding play button. Regarding the coloring of Fig. 5.7, it is visualized by an orange triangle. When activated, the corresponding *pause* button, *stop* button, *next* button and *previous* button control the machine-based modeling according to the use cases presented in Fig. 5.2.

Use Case—Initiate 3D-AR Simulation in the Augmentor:

According to the second level of the NEA designed (*Process Analysis and Simulation*, as shown in Fig. 4.8), the Augmentor visualizes the simulation results by anchoring the affected process modeling items within the real world. The simulation is accessed by activating the corresponding play button. Regarding the coloring of Fig. 5.7, it is visualized by a purple triangle. When activated, the corresponding *pause* button, *stop* button, *next* button and *previous* button control the ANN-based simulation, as shown in Fig. 5.2.

Use Case—Initiate Optimization in the Augmentor:

According to the third level of the NEA designed (*Process Optimization, Training and Testing*, as shown in Fig. 4.8), the Augmentor visualizes learning procedures by anchoring the affected process modeling items within the real world. The optimization can be accessed by activating the corresponding play button. Regarding the coloring of Fig. 5.7, it is visualized by a green triangle. When activated, the corresponding *pause* button, *stop* button, *next* button and *previous* button control the ANN-based optimization, as shown in Fig. 5.2.

5.3.2 Modelangelo Extension

With the intention to make the CoNM accessible to the traditional modeling world and considering the fact that the modeling tool called *Modelangelo* was selected by the CoNM-SESS (identified in section 6.1 and evaluated in section 7.1.1), the Modelangelo has been modified to execute a new functionality. Regarding the various activities (cf. NKMM based activities of section 4.5.1), Fig. 5.7 focuses on the realization of the use cases demanded by the CoNM software (section 5.1.2). The activity diagram presented here has thus guided the development of the Modelangelo's extensions.

Following a variant of the UML's activity diagram type, in the figure, one can see a sequence of activities that would make the workflow for CoNM-relevant activities

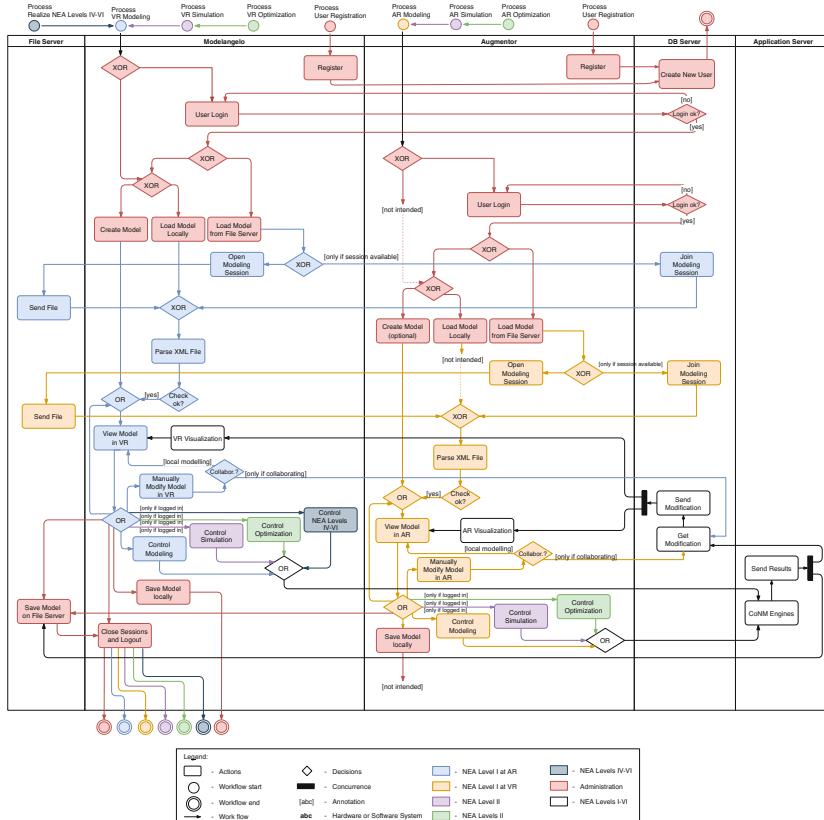


Figure 5.7 Workflow of CoNM Modeling (as UML Activity Diagram Variant)

clear. This includes questions regarding the tools that are accessed and realized. While each sequence of activities is represented by a rectangular shape with rounded corners, its colors refer to the corresponding NEA levels, as shown in Fig. 4.8. Further, its starting and ending points are visualized by circular shapes. As a variant, the operator or decision point has been characterized by textual annotations. Because of the use of swim lanes, which is a further variation of UML's activity diagram type, the corresponding hardware or software system carrying an activity becomes transparent.

Besides various administration processes, including the registration, login and logout procedures, as well as the access to new and old models, modeling can be realized from both, the Modelangelo and the Augmentor. This corresponds to the first NEA level and focuses on human-based modeling. While the workflow on the Modelangelo has been highlighted in blue, the corresponding workflow on the Augmentor has been presented in orange. Remarkably, the Augmentor's focuses on functioning as a visualization and information gathering tool; it does not intend to create a local file base. In distinction to this, the Modelangelo enables the operation on both the local side and server side.

Further, the Modelangelo and Augmentor enable the first three NEA levels (section 4.3); the enabling refers to the ANN-based modeling (see activity *Control Modeling*) and is controlled via the tools (blue and orange color), the simulation controlling-related activity (purple color), as well as the *Control Optimization* activity (green color). Because of the Augmentor's focus on providing shortcut functions, workflows issuing NEA levels IV–VI can only be controlled by the Modelangelo (colored in dark blue).

Regardless of the specific control instruction sent by either the Modelangelo or the Augmentor, this instruction will be realized by the corresponding CoNM engines. These refer to the software systems on the application server corresponding to Fig. 5.3. These kind of overarching activities have been highlighted in white. Further, the processing results are synchronized twofold: first with the corresponding database systems, so that a visualization of all kinds of devices enriches the current view of a model, and second with the file systems realizing the long-term result storing.

Using this arrangement of workflows, the bidirectional AR modeling was enabled within the Modelangelo, as it was conceptualized in section 5.1.1. Further, the workflows were harmonized with the architectural setup presented in section 5.2 so that the CoNM use case arrangement is reflected as it was presented in section 5.1.2. Since this figure does not issue the technically required, complex interplay of hardware systems, Fig. 5.8 focuses on the potential interactions demanded by the CoNM software. The sequence diagram presented here thus guided the implementations of the Modelangelo's extensions, such as corresponding server connections, instruction transfers and function calls.

Following a variant of the UML sequence diagram type, one can see the system interactions in time sequence in the figure . By this, it becomes clear which software and hardware components are required by the use case identified in section 5.1.2. The hardware and software components ("systems") have been visualized by UML's component shape and refer to the Modelangelo, Augmentor, database server, file server and application server; for instance, each component's lifeline has been visu-

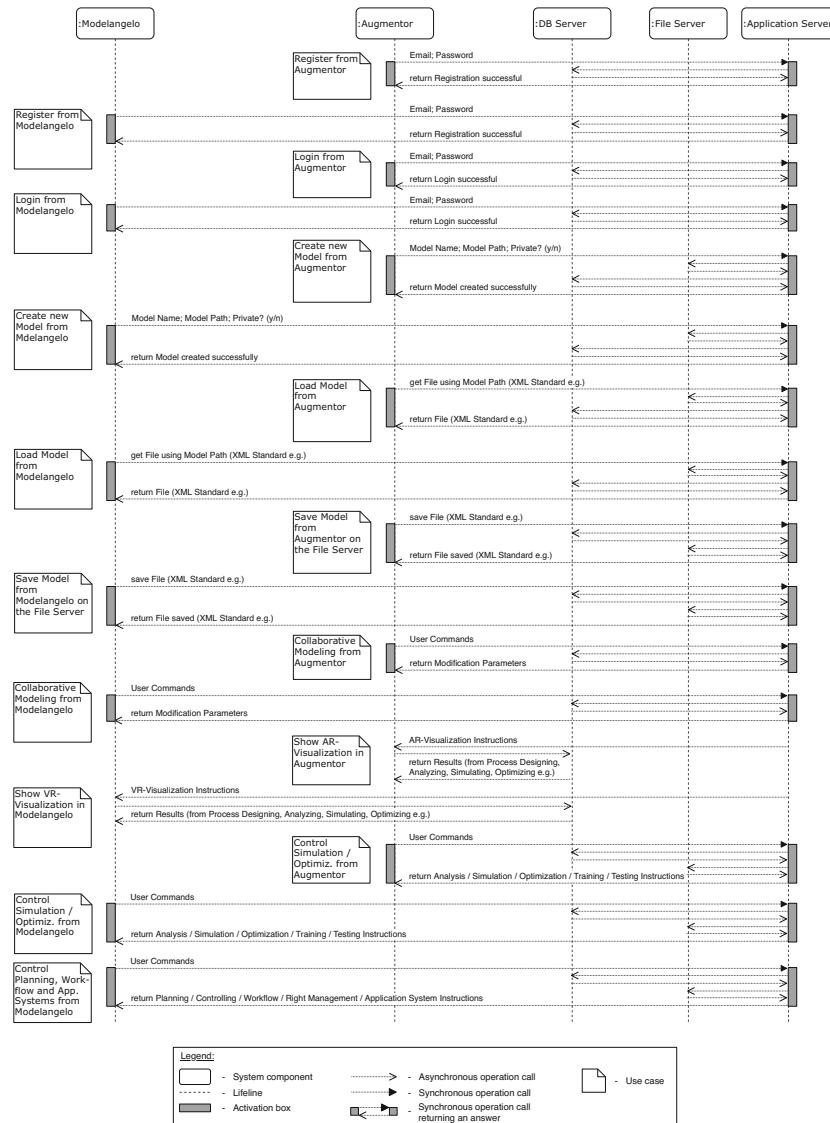


Figure 5.8 System Interactions Arranged in Time Sequence (as UML Sequence Diagram Variant)

alized by a dashed line. To showcase different kinds of interactions, the so-called message exchanges or operation calls have been visualized by horizontal arrows, and the course of time is shown by a vertical time axis from the top to the bottom. Synchronous interactions block the corresponding lifeline during call processing, and this is visualized by the gray activation box. Asynchronous interactions do not block a system's lifeline. Although the UML's sequence diagram type principally visualizes the flow through a decision tree of one individual system call, as a variant, the figure shows the flow through decision trees for the collection of use cases in Fig. 5.2. Hence, as a variant, use cases have been complemented by UML's well-known note symbol.

From this figure, one can recognize that the NPM-, NPS- and NPO-related use cases initiate an interaction sequence from both the Modelangelo as well as the Augmentor. This was required by the use cases identified (Fig. 5.2) and their workflows (Fig. 5.7). These thus refer to the NEA level I ("Process Design"), NEA level II ("Process Analysis and Simulation") and NEA level III ("Process Optimization, Training and Testing") described in section 4.3. As the NEA levels IV–VI can only be carried out by the VR-based modeling tool, the corresponding interactions can only be initiated by the Modelangelo.

Further, one can see that the interactions initiated either by the Modelangelo or the Augmentor follow the system structure of interactions. Principally, the application server is accessed by control instructions from the AR and VR devices, and it manages the communication flow asynchronously. Here, the operational databases are updated first, as it is a time-critical operation. File servers are updated after this, as the storage of simulation results, for instance, is not a time-critical operation.

Choice of Use Cases: Besides trivial use cases, such as the registration routine, the login process or the file loading and storing pattern, which have been implemented, the following only addresses the most relevant cases for the clarification and functional proof-of-concept of the CoNM. The NPM, NPS and NPO refer to the first three levels of the NEA designed (Fig. 4.8).

Use Case—Realize Modeling form Modelangelo: According to the first level of the NEA designed (*Process Design*), the Modelangelo visualizes the anchoring of the process modeling items within the virtual world. Modelangelo extensions thus issue a first use case variant—modeling by human model creators (modeling Def. 5). For this, modeling items are dragged and dropped from the modeling palette on the left to the modeling space of the center. This follows the traditional usability concept of the Modelangelo. Fig. 5.9 presents an overview of the extensions implemented. Concrete novelties are described thereafter.

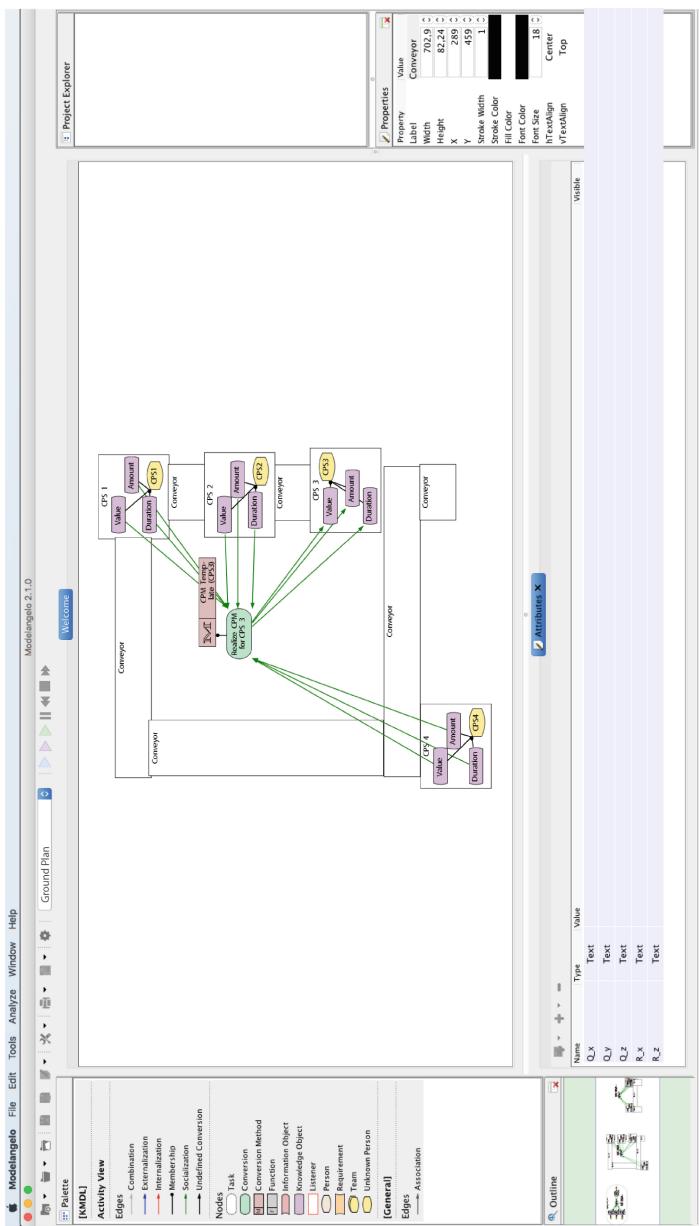


Figure 5.9 Modelangelo Extension—Ground Plan. (Example from Grum and Gronau, 2017)

By applying the NMDL meta-model to the Modelangelo by rebuilding Fig. 4.18 (atomic NMDL taxonomy views), Fig. 4.19 (complex NMDL taxonomy views) and Fig. 4.20 (flow-oriented NMDL views) within the Modelangelo’s language editor, the syntax specified in section 4.6.3 and the corresponding shapes of section 4.6.4 have been prepared for the model creation.

Corresponding to the perspectives, views and manifestations conceptualized in section 4.1 and specified in section 4.6, different kinds of models can be accessed from the *project explorer* on the software’s right, and the individual view manifestations can be accessed from the *selection panel* at the top of the software. By this, relevant perspectives, views and manifestations can be accessed in accordance with Fig. 4.14. So, for example, the sketch plan or ground plan of the activity view can be accessed, which prepares the visualization within the real world by the *Augmentor* (see Fig. 5.6). Fig. 5.9 presented here thus shows the ground plan of an Industry 4.0 knowledge generation. The corresponding sketch plan, reading-flow-optimized model as well as two further examples can be found at (Grum and Gronau, 2017).

Relevant parameters for the CoNM have been extended by the *attribute panel* at the software’s bottom as well as by the *properties panel* at the software’s right. Here, for example, one can find the element-wise interpretation of O , P , Q and R (section 4.6.1 and section 4.6.2). Further examples refer to the 3D shapes of objects, animations and AR target images.

Further, the Modelangelo extensions issue a second use case variant, which refers to the modeling done by machine-based model creators (modeling Def. 5), meaning ANN systems. This is accessed by activating the corresponding play button. Regarding the coloring of Fig. 5.7, it is visualized by a blue triangle. When activated, the corresponding *pause* button, *stop* button, *next* button and *previous* button control the machine-based modeling, as shown in Fig. 5.2.

Use Case—Initiate 3D-AR Simulation in Modelangelo: According to the second level of the NEA designed (*Process Analysis and Simulation*), the Modelangelo visualizes simulation results by their anchoring of affected process modeling items within the virtual world. The simulation is accessed by activating the corresponding play button. Regarding the coloring of Fig. 5.7, it is visualized by a purple triangle. When activated, the corresponding *pause* button, *stop* button, *next* button and *previous* button control the ANN-based simulation, as shown in Fig. 5.2.

Use Case—Initiate Optimization in Modelangelo: According to the third level of the NEA designed (*Process Optimization, Training and Testing*), the Modelangelo visualizes learning procedures by their anchoring of affected process modeling items within the virtual world. The optimization is accessed by activating the corre-

sponding play button. Regarding the coloring of Fig. 5.7, it is visualized by a green triangle. When activated, the corresponding *pause* button, *stop* button, *next* button and *previous* button control the ANN-based optimization, as shown in Fig. 5.2.

5.3.3 Transformation Tool Development

Regarding the various types of third party applications and servers (Fig. 4.8) and the intention to consider them via interfaces as systems giving an input for the CoNM, for instance, as symbiotic simulation systems or as real-time machine data generators, or receiving a CoNM output, such as external planning systems, external simulators or workflow engines, an engine called *Transformation Tool* has been set up as a new development. It considers XML files coming from the Modelangelo and translates them into third party formats.

The tool's major function is to transform the XML-based models coming from the Modelangelo or Augmentor into files required for programming libraries such as PyBrain or TensorFlow and vice versa. It enables the information exchange between the ANN structures constructed and their model mapping required by the NMDL (specified by section 4.6.3 and 4.6.4).

5.3.4 PyBrain Extension

Although both PyBrain and the combination of TensorFlow and LUCID have been selected as an attractive CoNM foundation (presented in section 6.1 and section 7.1.1), the following builds on extensions of the programming library called PyBrain. It shows how to make further libraries usable for the CoNM.

As basic and specialized ANN structures such as RNN and LSTM modules and trainers have been provided by PyBrain, extensions referred to implementations that realized a concept called *systematic exhaustion of Knowledge objects* (SEKO), as was required by the SEKO (section 4.7), as well as implementations that realized a concept called *learning principle of neuronal recursion* (LPNR), as was required by the LPNR (section 4.8.1).

For a technical basement for the CoNM, the PyBrain extension created is considered by any CoNM engine (Fig. 5.3). This corresponds to the libraries' contextual interpretation as follows. For example, it is interpreted as a modeling technique for NPM initiated by the *process design* engine. Alternatively, it is interpreted as a simulation technique for NPS initiated by the *simulation and analysis* engine, or the

interpretation considers the library as an optimization technique for NPO initiated by the *optimization* engine.

5.3.5 Interim Conclusion

The prototypes implemented here intend to carry the CoNM so that various kinds of ANN systems and models are integrated.

By implementing the Augmentor, AR devices are enabled to fulfill the requirements of the CoNM. Here, tempo-spatial relations can be accessed right within the reality so that the information quality for ANN input is increased due to a more precise modeling coming from the field of information occurrence. Further, the CoNM operation can be visualized within the field of operation.

Further, using the extension of the Modelangelo, VR devices are enabled to fulfill the requirements of the CoNM. Here, tempo-spatial relations can be accessed through various kinds of visualizations. So, a complex set of functions can be controlled, and a flexible view of the inner working on ANN systems can be realized.

The implementation of a transformation tool takes care of the preparation of various file formats so that the exchange among different kinds of CoNM tools is enabled. For example, XML files from the Modelangelo can be transformed into an interpretable format for the programming library called PyBrain to a 3D modeling tool such as NeuroConstruct (section 2.6.2.6), or simulation scenarios can be exported to third party applications such as NEURON (section 2.6.2.8).

Because of the extension of PyBrain, NMDL-based models are projected on a machine-interpretable algorithmic process. Through SEKO extensions, the machine-based model construction is prepared to be carried out as the NEA activity of the first level. Further, through LPNR extensions, the iterative and tempo-spatial precise ANN improvement is enabled.

Overall, as these prototypes jointly clarify how to carry out the NKMM-required activities (section 4.5) corresponding to the NEA architecture fields (Fig. 4.8), measures from the ordering system, as shown in Fig. 4.13, can be carried out systematically.



Demonstration

6

The demonstration considers the design set up shown in section 4 and examines if the requirements of this contribution (see section 3.1.3) can be addressed in practice. According to Peffers et al. (2007), this refers to the fourth step of a design-oriented research proceeding, and the section presents the use of sub-artifacts that will jointly manifest the CoNM. The following demonstrations intend to show the functioning of the CoNM with respect to the main research question (section 1.2). To this end, this section has been structured as follows:

The first sub-section presents the demonstration of the selection system being applied to all in all 38 objects of investigation (modeling languages, modeling and simulation tools, and deep learning libraries), the design of which can be seen in section 6.1. With this, the most attractive technical foundation for the construction of the CoNM will be identified. The second sub-section presents the demonstration of the NMDL based on theoretical examples. The validation of the joint CoNM will be demonstrated by the experiments described in the sections following thereafter.

6.1 Identification of an Appropriate Technical Foundation

Beside the qualitative evaluation of individual OoIs presented in section 2, the following applies the generic evaluation and selection system (GESS) design presented in section 4.2 and demonstrates its specific expression as CoNM-Specific Evaluation and Selection System (CoNM-SESS). As this system is demonstrated as a quantitative tool, the following is oriented to the empirical evaluation method (presented in

Supplementary Information The online version contains supplementary material, such as Appendix A, B, C and D whose references are indicated in the text, available at https://doi.org/10.1007/978-3-658-35999-7_6.

section 3.2.4). It has been structured in accordance with its phases, and it requires nine steps.

6.1.1 Problem Specification

The first step of the empirical evaluation method was to define the research problem. For the demonstration purpose, this refers to the question as to how the GESS designed can be applied to the context of the ANN process domain so that 1) a technical foundation for the CoNM construction can be identified and 2) the CoNM can be evaluated on an equal footing with contemporary concepts. As a qualitative evaluation has already been carried out by the SLR presented in section 2.7 and a research gap has been identified on a qualitative level, the quantitative characterization of the population of OoIs coming from the process domain as well as from the ANN domain have been focused upon.

6.1.2 Survey Planning and Preparation

The second step of the empirical evaluation method was to construct the survey instrument. For this, first, the evaluation aspects and second objects of investigation were specified. These solidify the $m_{i,j}$ (aspects) and o_k (OoI) designed in section 4.2.

Specification of evaluation aspects: The evaluation criteria have been derived from the generic CoNM meta-model designed in section 4.1.3. Here, the focus was on the availability of each kind of modeling object represented in Fig. 4.4. Note that each gray rectangle in this figure is connected to one evaluation aspect $m_{i,j}$ of Eq. 4.1. Since 132 modeling objects are grouped by nine modeling perspectives, $n_i = 9$. Since the meaning of modeling objects has already been explained in section 4.1.3, and the availability of each approach following the operationalization of the GESS is trivial, further explanations are not required. Therefore, we have the business process context perspective with $n_{1,j} = 16$, the organizational perspective with $n_{2,j} = 12$, the behavioral perspective with $n_{3,j} = 17$, the functional perspective with $n_{4,j} = 11$, the informational perspective with $n_{5,j} = 15$, the knowledge perspective with $n_{6,j} = 10$, the simulation perspective with $n_{7,j} = 10$, the neuronal perspective with $n_{8,j} = 18$, and the process ANN perspective with $n_{9,j} = 23$.

Three further evaluation perspectives have been identified when considering the requirements derived from theoretical foundations for modeling and simulation tools (section 2.6.1) and deep learning tools (section 2.6.2). Here, again, the focus was on

the availability of each kind of technical concept. Around 42 tool aspects are grouped by 3 tool-specific evaluation perspectives, so we have here: $n_i = 9 + 3 = 12$ as well as $n_{10,j} = 20$ for modeling tool aspects, $n_{11,j} = 10$ for simulation tool aspects, and $n_{12,j} = 22$ for ANN and DL tool aspects. Since the requirements and concepts have already been explained within the theoretical foundation, and the availability of each follows the operationalization of the GESS, further explanations have been deemed unnecessary.

Selected aspects can be found in Appendix A.2–A.8. Here, the individual aspect, $m_{i,j,k}$, can be found as the title for any row. Any superscript visualized indicates the origin of the presented aspect, $m_{i,j,k}$, and has been specified in Appendix A.9. For this, 210 sources have been considered.

Specification of objects of investigation: Objects of investigation can be found in state-of-the-art tools and concepts presented within the theoretical foundations, considered as follows:

Four (business) process modeling languages were considered as they were selected in section 2.1.3. Three knowledge management concepts and modeling languages were considered as they were selected in section 2.2.3 and 2.2.2. Three simulation modeling languages (cf. section 2.3.3) and three ANN modeling languages were considered (cf. section 2.5.9). The selection maxim was bound to the following priorities:

1. Modeling languages having a broad market distribution
2. Modeling languages having a broad acceptance in the modeling community
3. Modeling languages being applied in at least one scientific publication

Following the recommendation of Scheer (2002, p. 132), isolated architecture discussions without tool support or method discussions without tool support were dismissed since designing an efficient practice is impossible. Up to here, 13 modeling languages can be identified as objects of investigation and $n_k = 13$.

Further, four best practices (cf. section 2.4.2) and four organizational learning concepts have been considered (cf. section 2.4.3). Up to here, 13 modeling languages can be identified as objects of investigation and $n_k = 21$.

Additionally, four modeling and four simulation tools, which were discussed in section 2.6.1, were considered. A selection maxim was bound to the following priorities:

1. Top three tools of Gartner's most current market research study (Dunie et al., 2017)

2. All tools of the Market Research Study of Fraunhofer IAO (2014) that satisfy CoNM required dimensions such as modeling, analysis, simulation, process execution (Drawehn et al., 2014).
3. Modelangelo as department specific tool.
4. (Tools being applied in at least one scientific publication.)

Following the recommendation of Scheer (2002, p. 132), isolated graphical tools (e.g. MS Visio, OmniGraffle) that do not support architectural perspectives or modeling methods were dismissed since creating an efficient practice design with them is impossible. Here, five deep learning programming libraries and four ANN tools focusing on simulation and 3D were considered as objects of investigation, which have already been discussed in section 2.6.2. Here, Scheer's recommendations were relaxed because of their focus on neuronal learning concepts. A selection maxim was bound to the following priorities:

1. Tools having a broad market distribution
2. Tools having a broad acceptance in the ANN community
3. Tools being applied to at least one scientific publication

All together, 35 objects of investigation were identified, so $n_k = 35$.

Selected OoIs can be found in Appendix A.2–A.8. Here, the individual OoIs o_k can be found as the title for any column. Any superscript visualized indicates the origin of the presented objects, o_k , and has been specified in Appendix A.9. For this, 210 sources have been considered.

Further, as the CoNM-SESS is intended to evaluate the CoNM constructed on an equal footing with the concepts evaluated, the CoNM itself stands as a separate OoI. Then, all together, 36 OoIs can be identified ($n_k = 36$).

Survey form: The third step of the empirical evaluation method was to define the survey form. According to Häder (2010, p. 187), an interview form was distributed via the Internet to gather a broad range and varied group of people could be tested. As an online survey form, information was digitally collectable, which is attractive for an efficient data processing.

Sampling procedure: The fourth step of the empirical evaluation method was to describe the sampling procedure. This includes describing the test person composition, as well as the extent and period of the survey (Diekmann, 2012, p. 186–230). Surveyed experts were selected from the process domain, which includes business process experts, knowledge management experts, and simulation experts, as well as

the ANN domain, which includes ANN programming experts, business analytics experts, and neuroscience experts. These experts were given the digital survey form and were asked to complete the survey within a period of six months. This gave the opportunity to conduct research on the objects of investigation. As questions arose, for instance, regarding the formulation, insecurities regarding the operationalization, etc., experts were consulted either over telephone conferences or in person in order to resolve the problem.

Pretest: The fifth step of the empirical evaluation method was to carry out a pretest. After a few test persons were surveyed, the survey form was discussed individually. Feedback points were collected regarding a reformulation of some survey questions, a more comprehensible formulation of some aspect descriptions, and the transparent visualization of evaluations inserted. The latter was visualized by circular elements in initial survey versions, which had lead to confusion about the answer state. Hence, this form of visualization was only used for the result presentation.

6.1.3 Data Collection

The sixth step of the empirical evaluation method was to use the tested survey instrument for the purpose of data collection. The survey instrument was accompanied with an introduction letter clarifying the research objective. As the CoNM-SESS was to be used for the CoNM evaluation, further, a presentation including textual description for the CoNM was included. If questions arose, a workshop-based education guaranteed a joint comprehension.

A quality assurance considered the following: First, if any answers were missing, test persons were asked to fill in the missing entries. Particularly in this step, insecurities and questions became transparent. Second, if any answers referred to invalid values, test persons were asked to correct those entries. Third, if any of the survey forms was not submitted, the respective test person was individually contacted. As the survey was carried out over a long period of six months, some had simply forgotten to carry out the evaluations. Others were busy with their own work. If welcomed, an individual extension of the submission period was granted.

In total, a test person base of 85 entries has been realized.

6.1.4 Data Analysis

The seventh step of the empirical evaluation method was to create an analyzable data file per test person. A Python-based scripted, implemented for the analysis, read the file collection, implemented the GESS designed in section 4.2. and so

generated KPI-based visualization designs, which had been stored as xls file, so that spreadsheet programs such as Excel are able to deal with the files generated and formatting issues for the print as pdf files.

Answers not lying within the permitted scale were clubbed under the category of “out of range”. Missing values were clubbed under the category of “no information”. As these were inconsistent data, they were regarded useless for analysis purposes (Schnell, Hill, and Esser, 2011, p. 9).

The eighth step of the empirical evaluation method was to carry out the statistical data analysis. As Fig. 4.5 visualizes, the analysis considered three quantitative survey bases. Each of them individually refers to one demonstration of the CoNM-SESS and is considered separately in the following.

6.1.4.1 Personal Expertise

As the first demonstration of the CoNM-SESS (Fig. 4.5), the personal expertise is based on the evaluation of the author of this contribution. It has been built based on the description and qualitative evaluation of each OoI described in section 2. The detailed view on the evaluations for each individual aspect $m_{i,j,k}$ can be found in Appendix A.2–A.8. Any superscript visualized indicates the origin of the presented aspects, $m_{i,j,k}$, as well as of objects, o_k , and has been specified in Appendix A.9. Its overview has been presented in the following (Fig. 6.1) and can be found as an enlarged version in Appendix A.1.

In this visualization, one can find $V_{i,k}^{\mu,\mp}$ at each inner cell element, and $R_k^{\mu,\mp}$ can be found under each column. Regarding the color map ranging from red to green, one can identify the following:

First, apparently, no OoI satisfies the perspectives identified all-embracingly. OoIs showing the highest $R_k^{\mu,\mp}$ values refer to the “Potsdam Knowledge Management Model and the KMDL” with $R_7^{\mu,\mp} = 28$. On the second place, one can find the tool called “ARIS Platform” ($R_{20}^{\mu,\mp} = 25$), and on the third place, the tool called “Modelangelo” with $R_{22}^{\mu,\mp} = 24$. OoIs having the smallest $R_k^{\mu,\mp}$ refer to the “Flow Charts” ($R_{10}^{\mu,\mp} = 12$), the “NeuroLOITs and NeuroTessMesh” ($R_{30}^{\mu,\mp} = 12$), as well as to the “SimPy” library with $R_{25}^{\mu,\mp} = 12$.

Second, the “ANN modeling and DL tools” are not capable of being extended regarding the perspectives called the “business process context perspective”, the “organizational perspective”, the “behavioral perspective”, the “functional perspective”, the “informational perspective” and the “knowledge perspective”. Values here mainly refer to $V_{1\dots6,27\dots35}^{\mu,\mp} = 1$. While their focus on the “neuronal perspective” becomes clear ($V_{8,27\dots35}^{\mu,\mp}$ range from 1.7 to 3.0), the criteria for the “simulation perspective” as well as for the “process ANN perspective” are evaluated to be adequate for an effortful extension ($V_{7,27\dots35}^{\mu,\mp}$ and $V_{9,27\dots35}^{\mu,\mp}$ are around 2.0).

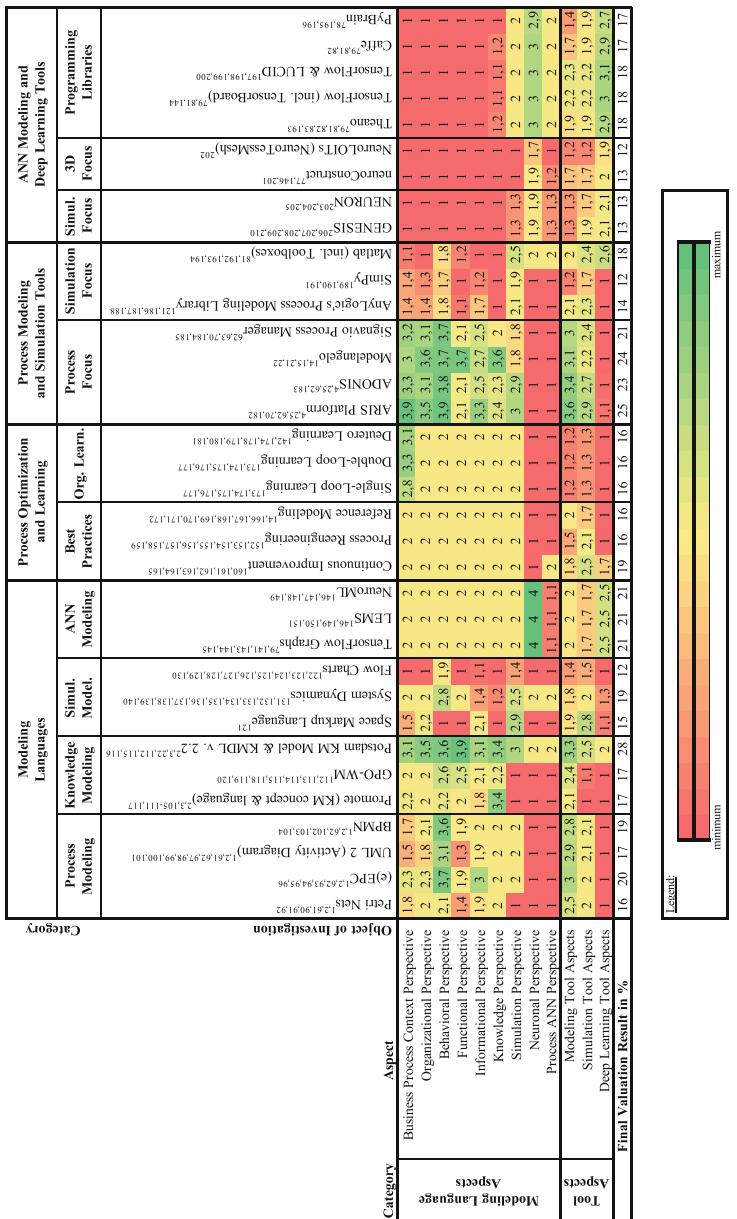


Figure 6.1 The Demonstration of the Selection System, Evaluation of Marcus Grum

Third, in contradiction to “ANN modeling and DL tools”, tools from the category named “process modeling and simulation”, which have a focus on processes, establish a contradictory characterization. These show strengths in the “business process context perspective”, the “organizational perspective”, the “behavioral perspective”, the “functional perspective”, the “informational perspective” and the “knowledge perspective” and have aggregated mean values $V_{1\dots7,20\dots23}^{\mu,\mp}$ from 1.8 up to 3.7. Further, they are not appropriate for the “neuronal perspective” and “process ANN perspective”. Here, they provide aggregated mean values $V_{6,20\dots23}^{\mu,\mp}$ and $V_{9,20\dots23}^{\mu,\mp}$ referring to 1.0. Aside, considering tools having a simulation focus, these establish the same characterization but have weaker manifestations than tools with process focus.

Fourth, neither tools from the cluster “process and knowledge modeling languages” ($V_{8-9,1\dots10}^{\mu,\mp}$ refer to 1.0) nor tools from the “process optimization and learning tools” cluster ($V_{8-9,14\dots25}^{\mu,\mp}$ refer to 1.0) are capable of being extended for the perspectives called the “neuronal perspective” and the “process ANN perspective”. At least for the “neuronal perspective”, the ANN modeling languages examined can be a foundation ($V_{8-9,11\dots13}^{\mu,\mp}$ range from 2.1 up to 4.4), which would be complemented by the technical mechanisms of ANN programming libraries ($V_{8-9,31\dots35}^{\mu,\mp}$ range from 1.0 up to 3.0) and ANN tools ($V_{8-9,27\dots30}^{\mu,\mp}$ range from 1.0 up to 1.7).

Fifth, OoIs aside from the ANN domain are not extendable in regard with “DL tool aspects”. Their aggregated mean values $V_{12,1\dots10}^{\mu,\mp}$ and $V_{12,14\dots25}^{\mu,\mp}$ refer to 1.0. Here, ANN programming libraries show strengths, and tools from the “ANN modeling languages” cluster as well as the “KMDL” can stand as a foundation with effortful extensions ($V_{12,11\dots13}^{\mu,\mp}$ and $(V_{812,26\dots35}^{\mu,\mp}$ range from 1.9 up to 3.1).

Sixth, OoIs from the clusters called “process optimization and learning” as well as “ANN modeling” are adequate for the effortful extension of the perspectives called the “business process context perspective”, the “organizational perspective”, the “behavioral perspective”, the “functional perspective”, the “informational perspective” and the “knowledge perspective”. They majorly show aggregated mean values ($V_{1\dots7,11\dots19}^{\mu,\mp}$ around 2.0).

As the variance for one test person refers to zero, the corresponding $V_{i,k}^{\sigma^2,\mp}$ and $R_k^{\sigma^2,\mp}$ have not been established.

6.1.4.2 Expert View

As a second demonstration of the CoNM-SESS (Fig. 4.5), the expert view is based on the evaluation of the specialists being surveyed. Hence, they will relativize the evaluations from the author. The aggregated mean $V_{i,k}^{\mu,\pm}$ is visualized in Fig. 6.2. Its enlarged version can be found in Appendix A.10.

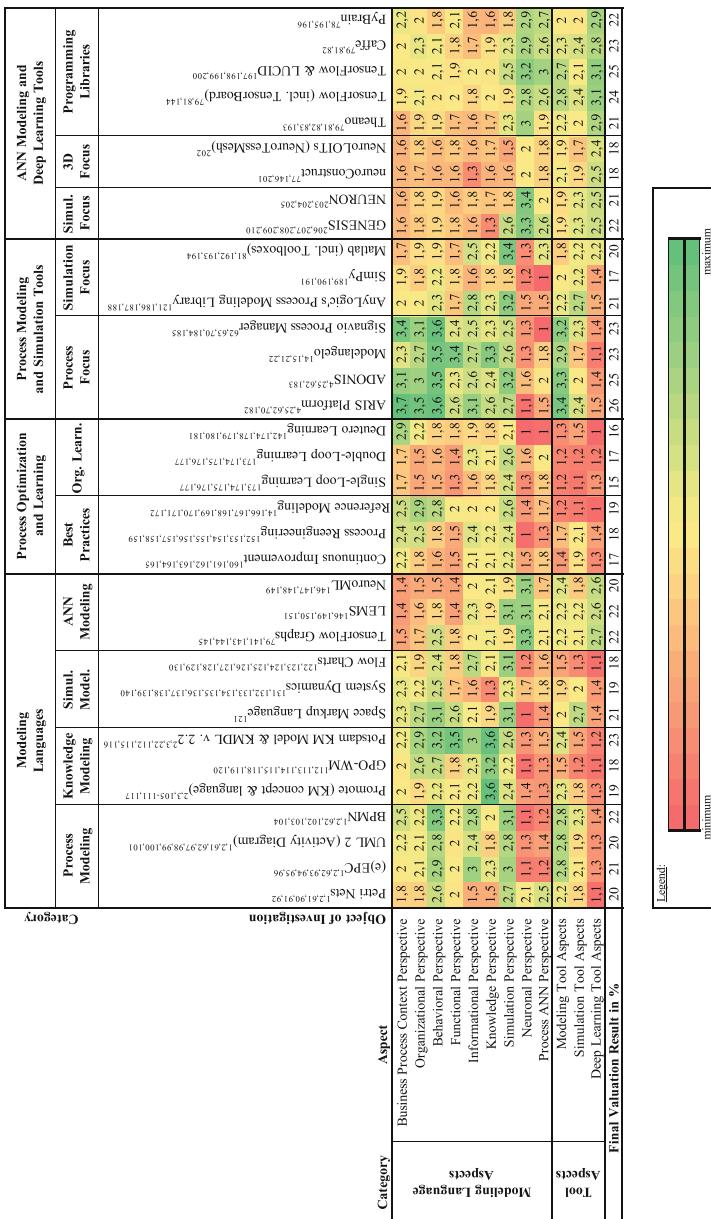


Figure 6.2 The Demonstration of the Selection System, Evaluation of Experts (Mean)

According to the insights presented in section 6.1.4.1, relativizations can be found as follows:

First, the broad expert view confirms non-OoI to satisfy the perspectives identified in an all-embracing manner. Under the first six positions having the highest $R_k^{\mu,+}$, one can confirm the “Potsdam Knowledge Management Model and the KMDL” ($R_7^{\mu,+} = 23$), the “ARIS Platform” ($R_{20}^{\mu,+} = 26$) and the “Modelangelo” ($R_{24}^{\mu,+} = 23$) as dominant. Further, the programming libraries around “TensorFlow” ($R_{32-33}^{\mu,+}$ range from 24 up to 25) and “Caffe” ($R_{34}^{\mu,+} = 23$) have been identified as attractive foundations. The weakest tool is the “Single-Loop Learning”, with aggregated mean values $R_{17}^{\mu,+} = 15$. The weakest tools identified by the author of this contribution have been confirmed, as they can be found among the weakest seven tools.

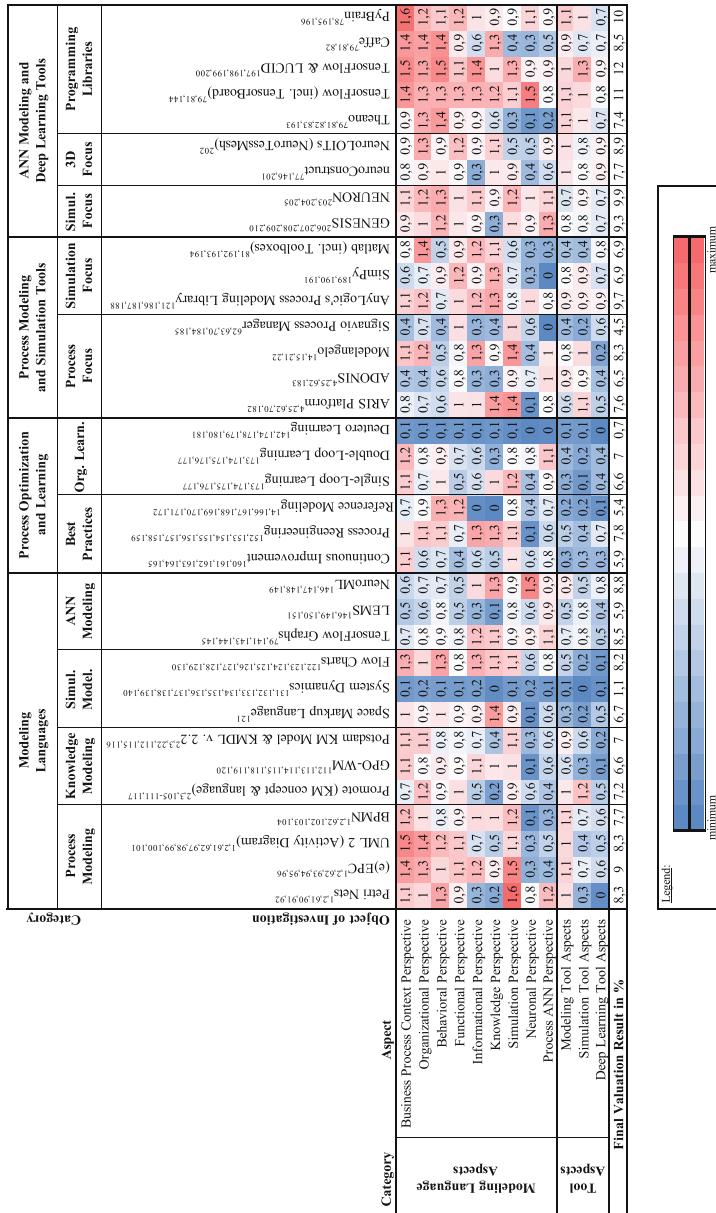
Although insights have to be relativized because of more or less slight deviations, the points two, three, four and five have been confirmed by the broad expert base.

Sixth, the impression of the OoIs from the clusters called “process optimization and learning” as well as “ANN modeling” is more differentiated than by the author of this contribution. With only slight deviations, values are majorly around the evaluation of requiring effortful extensions for the perspectives called the “business process context perspective”, the “organizational perspective”, the “behavioral perspective”, the “functional perspective”, the “informational perspective” and the “knowledge perspective”. They majorly show aggregated mean values ($V_{1...7,11...19}^{\mu,+}$ around 2.0). Looking at this, it can be said that the author’s impression has been confirmed.

Regarding the mean identified before, the agreement of the broad expert base becomes clear with the aid of the variance $V_{i,k}^{\sigma^2,+}$. Its overview can be seen in Fig. 6.3. Further, the enlarged version can be found in Appendix A.11.

According to the mean presented before, a broad agreement (small variance values) or a broad difference in evaluations (big variance values) can be found as follows:

First, the expert base shows a broad agreement regarding the tools “System Dynamics” and “Deutero-Learning”. With $R_9^{\sigma^2,+} = 1.1$ for “System Dynamics” and $R_{19}^{\sigma^2,+} = 0.7$ for “Deutero-Learning”, evaluations deviated about one point over all OoIs. Apparently, here, the documentation has been best prepared, the tools are well known or their potential has been made very clear, because of which the broad expert base is certain regarding this evaluation. A worse agreement can be identified with the ANN programming libraries around “TensorFlow” ($R_{32..33}^{\sigma^2,+} = 1.1$ range from 11 up to 12) and “PyBrain” ($R_{35}^{\sigma^2,+} = 10$). Apparently, the evaluation is more



Least:

maximum

difficult, for instance, because of a more challenging interpretation of ANN tools within the process domain, which makes the need for a joint concept such as the CoNM clear. Further, the best and worse OoIs identified so far provide average variance values, which speaks for an evaluation without any conspicuousness.

Second, regarding the “ANN modeling and DL tools” cluster identified in section 6.1.4.1, rather high variances $V_{1\dots6,27\dots35}^{\sigma^2,+}$ can be identified. This results in the complex ANN tool evaluation mentioned before.

Third, regarding the “process modeling and simulation” cluster identified in section 6.1.4.1, rather small variances $V_{1\dots7,20\dots23}^{\sigma^2,+}$ can be identified, which speaks for a clear positioning of process model tools and a well-accessed tool process modeling tool market. Regarding slightly higher variances for the tools having a simulation focus ($V_{1\dots7,24\dots26}^{\sigma^2,+}$), a slightly worse positioning of process simulation tools and a slightly worse accessed process simulation tool market can be identified.

Fourth, the expert base apparently uniformly evaluates the perspectives called the “neuronal perspective” and the “process ANN perspective” (small $V_{8..9,k}^{\sigma^2,+}$ values). Apparently, here, an evaluation is trivial because perspectives are not satisfied accordingly; particularly, tools from the process modeling domain show very small variance values ($V_{8..9,1\dots10}^{\sigma^2,+}$ and $V_{8..9,14\dots26}^{\sigma^2,+}$ range from 0.1 up to 1.2).

Fifth, apparently, the evaluation of process modeling tools in regard with “Deep Learning Tool Aspects” is simple, which can be seen at low variance values for tools from the process domain ($V_{12,1\dots10}^{\sigma^2,+}$ and $V_{12,14\dots25}^{\sigma^2,+}$). This might be because of the clear non-compliance of tools from the process domain.

Sixth, regarding OoIs from the clusters called “process optimization and learning” as well as “ANN modeling” identified in section 6.1.4.1, majorly, one can identify a broad agreement of experts (majorly small aggregated variance values $V_{1\dots7,11\dots19}^{\sigma^2,+}$). Apparently, here, the criteria fulfillment is clearer than the cluster around “Process Modeling” ($V_{1\dots7,1\dots4}^{\sigma^2,+}$), “Knowledge Modeling” ($V_{1\dots7,5\dots7}^{\sigma^2,+}$) and “Simulation Modeling” ($V_{1\dots7,8\dots10}^{\sigma^2,+}$). This might lie on the more complex evaluation of tools providing differences majorly in details. Here, the meaning for a qualitative analysis becomes clear and has been conducted beforehand, as shown in Fig. 4.5.

6.1.4.3 Popularity and Attractiveness

In the third demonstration of the CoNM-SESS (Fig. 4.5), the popularity is based on the evaluation of all test persons. This includes both kinds of test persons, experts as well as non-experts. The mean $V_{i,k}^{\mu,\pm}$ is visualized in Fig. 6.4. Its enlarged version can be found in Appendix A.12.

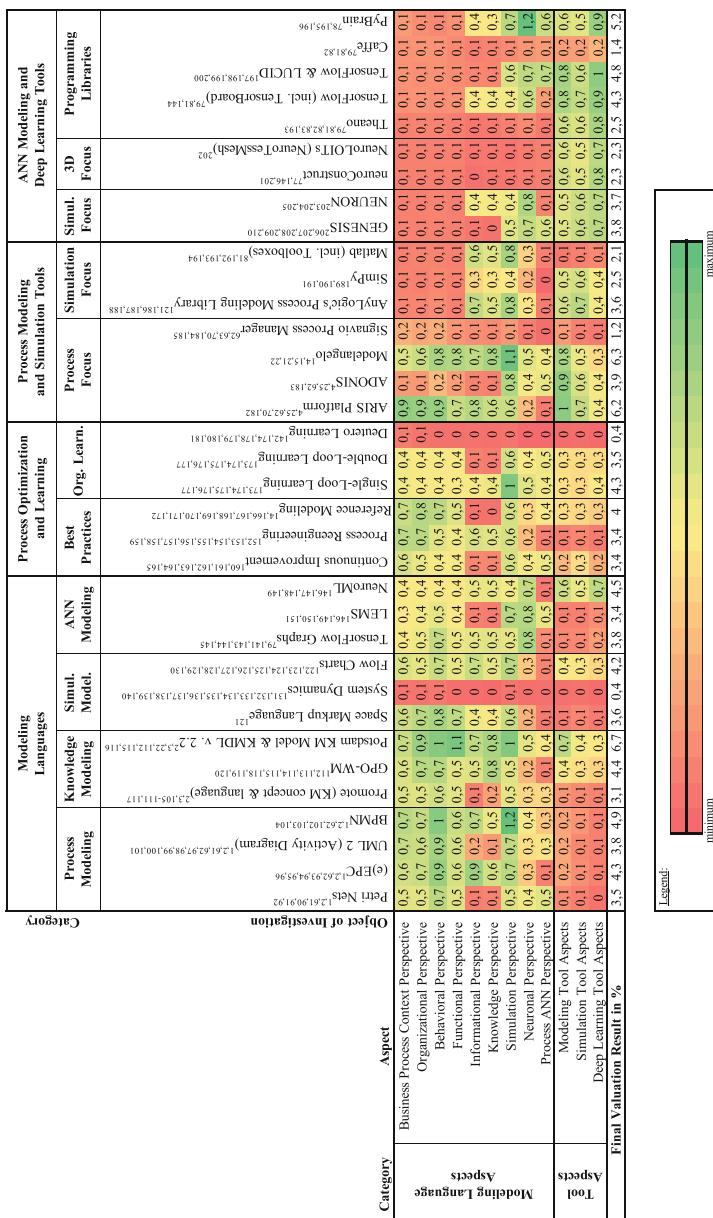


Figure 6.4 The Demonstration of the Selection System, Evaluation of all Test Persons (Mean)

According to insights presented in section 6.1.4.1 and section 6.1.4.2, relativizations can be found as follows:

First, the experts and non-experts jointly tend to not know about the tools “System Dynamics” and “Deutero-Learning”. With $R_9^{\mu,\pm} = 0.4$ for “System Dynamics” and $R_{19}^{\mu,\pm} = 0.4$ for “Deutero-Learning”, evaluations deviated from the expert-based evaluations. Apparently, although the documentation has been prepared well, the tools are well known, and their potential becomes very clear, this became clear because of the variance in the expert base (section 6.1.4.2); maybe the license model of “System Dynamic” is not attractive and the concept of “Deutero-Learning” has not been distributed widely. Further, the process modeling and simulation tool called “Signavio” and the ANN programming library called “Caffe” show bad mean values, with $R_{23}^{\mu,\pm} = 1.2$ and $R_{34}^{\mu,\pm} = 1.4$. The best mean over all criteria can be identified with the tools “Potsdam KM and KMDL” ($R_7^{\mu,\pm} = 6.7$), “Modelangelo” ($R_{22}^{\mu,\pm} = 6.3$) as well as the “ARIS Platform” ($R_{20}^{\mu,\pm} = 6.2$). Apparently, these tools are widely known based on the test persons surveyed.

Second, regarding the “ANN modeling and DL tools” cluster identified in section 6.1.4.1, rather low mean values $V_{1\dots6,27\dots35}^{\mu,\pm}$ can be identified. This might result because of the complex ANN tool evaluation identified in section 6.1.4.2.

Third, regarding the “process modeling and simulation” cluster identified in section 6.1.4.1, a heterogeneous characterization can be identified. Rather low mean values $V_{1\dots6,21}^{\mu,\pm}$ can be identified for “ADONIS” and for the “Signavio Process Manager” ($V_{1\dots7,23}^{\mu,\pm}$), mainly around 0.1. Best candidates identified (see first point) show better values $V_{1\dots7,20}^{\mu,\pm}$ and $V_{1\dots7,22}^{\mu,\pm}$. Here, the popularity of the OoI becomes transparent.

Fourth, the non-expert base confirms the expert base for the perspectives called the “neuronal perspective” and the “process ANN perspective”. These show low mean values $V_{8\dots9,1\dots26}^{\mu,\pm}$ ranging from 0 up to 0.5. This indicates that most test persons stated that they now knew the corresponding criteria fulfillment.

Fifth, the non-expert base confirms the expert base for the “Deep Learning Tool Aspects”. Here, one can find lower $V_{12,1\dots25}^{\mu,\pm}$ values for tools from the process domain and ANN modeling languages. This might be because of the clear non-compliance of tools from the process domain as well as modeling languages in general.

Sixth, regarding OoIs from the clusters called “process optimization and learning” as well as “ANN modeling” identified in section 6.1.4.1, the broad base of test persons confirmed the evaluation identified by the author (section 6.1.4.1) and by the expert-base (section 6.1.4.1), which becomes clear because of higher mean values $V_{1\dots7,11\dots19}^{\mu,\pm}$ referring to the average.

Regarding the mean value identified before, which issued the popularity and attractiveness of the CoNM construction, the agreement of all test persons becomes clear with the variance $V_{i,k}^{\sigma^2,\pm}$. Its overview is visualized in Fig. 6.5. Further, its enlarged version can be found in Appendix A.13.

According to the mean presented before, a broad agreement (small variance values) or a broad difference in evaluations (big variance values) can be found as follows:

First, the experts and non-experts jointly tend to evaluate the OoIs called “System Dynamics” ($R_9^{\sigma^2,\pm} = 0.9$) and “Deutero-Learning” ($R_{19}^{\sigma^2,\pm} = 0.7$) uniquely. Here, a broad agreement becomes transparent that confirms the expert-based evaluation presented in section 6.1.4.1. The worst agreement can be found regarding the tools called “ARIS Platform” ($R_{20}^{\sigma^2,\pm} = 16$) and “Potsdam KM Model and KMDL” ($R_7^{\sigma^2,\pm} = 14$), which have been identified as most well-prepared ($R_{20}^{\mu,+} = 26$, $R_7^{\mu,+} = 23$) and most attractive ($R_{20}^{\mu,\pm} = 6.2$, $R_7^{\mu,\pm} = 6.7$). Apparently, the best candidates do not establish a broad agreement on evaluations.

Second, regarding the “ANN modeling and DL tools” cluster identified in section 6.1.4.1, rather low variances $V_{1...6,27...35}^{\sigma^2,\pm}$ can be identified. These are in contrast to rather high values from experts $V_{1...6,27...35}^{\sigma^2,+}$ which might lie on the lacking attitude of non-experts to deal with technical details.

Third, regarding the “process modeling and simulation” cluster identified in section 6.1.4.1, the heterogeneous characterization identified by all test persons has been confirmed. While the best tools tend to have rather high variance values ($V_{1...7,20}^{\sigma^2,\pm}$ and $V_{1...7,22}^{\sigma^2,\pm}$ ranging from 1.3 up to 2.8), others show rather small variances ($V_{1...6,21}^{\sigma^2,\pm}$ and $V_{1...7,23}^{\sigma^2,\pm}$ ranging from 0.2 up to 0.8). Here, the popularity of the OoI becomes transparent.

Fourth, the evaluation of all test persons resembles the evaluation of the expert base, as they apparently uniformly evaluate the perspectives called the “neuronal perspective” and the “process ANN perspective” (small $V_{8...9,k}^{\sigma^2,\pm}$ values). Because of small mean values ($V_{8...9,k}^{\mu,\mp}$ and $V_{8...9,k}^{\mu,+}$), the need for a joint concept such as the CoNM becomes apparent, as different bases commonly identify the corresponding criteria to be disregarded by tools so far.

Fifth, the impression of the expert base is confirmed by the all test person base as the evaluation of process modeling tools in regard with “Deep Learning Tool Aspects” provides low variance values for tools from the process domain ($V_{12,1...10}^{\sigma^2,\pm}$ and $V_{12,14...25}^{\sigma^2,\pm}$ ranging from 0.1 up to 0.6). This supports the idea of non-compliance

Figure 6.5 The Demonstration of the Selection System, Evaluation of all Test Persons (Variance)

of tools from the process domain. Further, tool aspects establish a similar characterization with aggregated variance values $V_{10...19,1...10}^{\sigma^2,\pm}$ ranging from 0 up to 0.6 with individual exceptional values. Only software tools provide higher variance values $V_{10...12,20...35}^{\sigma^2,\pm}$. Apparently, higher variance values can be reasoned with the more complex evaluation of the software tool criteria fulfillment by software tools.

Sixth, the insights of the experts has been confirmed. Variance values $V_{1...7,11...19}^{\sigma^2,\pm}$ mostly show average values so that an ordinary evaluation becomes apparent.

6.1.5 Reporting

As the ninth step of the empirical evaluation method selected, the use of insights generated are focused upon. Following the process for a systematic selection, as shown in Fig. 4.5, insights generated by the SLR and evaluation on behalf of Bloom's Taxonomy, were professionalized by three expert bases presented in section 6.1.4.1, section 6.1.4.2 and section 6.1.4.3, so that the following presents a coherent facit and tool selection for the CoNM foundation.

Aside the fact that all OoIs considered are attractive to be called the foundation for the CoNM construction because each consideration increases the access to the ANN process domain established by this, some OoIs have been highlighted to be considered foundations. These can be complemented by further OoIs and concepts issued by the theoretical foundation so that the weaknesses of individual OoIs are eliminated and they can be considered individual progresses aside the CoNM-based progress.

The best candidates of the three bases analyzed are indicated by $R_k^{\mu,\mp}$, $R_k^{\mu,+}$ and $R_k^{\mu,\pm}$. No OoI satisfies the perspectives and aspects all-embracingly. With this, the research gap identified by the SLR (section) has been confirmed by the CoNM-SESS in three demonstrations.

As a modeling language and knowledge management concept, the “Potsdam KM Model and KMDL” issues process modeling perspectives best because of the highest mean values $V_{1...7,7}^{\mu,\mp}$, the second best broad expert evaluation $V_{1...7,7}^{\mu,\mp}$, and the best evaluation by all experts $V_{1...7,7}^{\mu,\pm}$. Concepts provided by the OoI called “ARIS Platform” here can suit as a fruitful complement. This is indicated by the high mean values $V_{1...7,20}^{\mu,\mp}$, the best broad expert evaluation $V_{1...7,20}^{\mu,+}$, and the second best evaluation of all experts $V_{1...7,20}^{\mu,\pm}$. The modeling software tool called “Modelangelo”

is particularly attractive for the implementation of CoNM concepts designed. Aside of its second best mean values $V_{1\dots7,22}^{\mu,\mp}$, its third best mean values $V_{1\dots7,22}^{\mu,+}$ and its third best mean values $V_{1\dots7,22}^{\mu,\pm}$ facing the OoI's Open Source license, its available for extensions.

Focusing on the perspectives called the “neuronal perspective” and the “process ANN perspective”, the ANN modeling language called “TensorFlow Graphs” is attractive (best $V_{8..9,11}^{\mu,\mp}$ values, best $V_{8..9,11}^{\mu,+}$ values and high $V_{8..9,11}^{\mu,\pm}$ values). The OoI called “LEMS” and “NeuroML” are attractive in terms of technical supplementation of modeling entities (see qualitative OoI analyses of corresponding OoIs in section 2.5.9.3) and cross-simulator application (cf. section 2.5.9.2). Further, the OoI called “Space Markup Language” is attractive in terms of the spatial modeling of simulation elements (cf. section 2.3.3.1).

Considering ANN mechanisms, the best candidates can be identified around the programming libraries of “TensorFlow and LUCID” (best $V_{12,33}^{\mu,\mp}$, best $V_{12,33}^{\mu,+}$ and best $V_{12,33}^{\mu,\pm}$), and the most popular and attractive ANN tool form the perspective of all test persons called “PyBrain” (best $R_{35}^{\mu,\pm}$). As these programming libraries are integrated with a standardized modeling coding, such as by LEMS or NeuroML, and a joint interpretation can be established because of the CoNM, the following can be enabled: first, ML-based ANN techniques can be visualized by 3D realistic visualizations (enabled by the OoI called “NeuroConstruct” and “NeuroTessMesh”); second, Modelangelo's extension called “Augmentor” can position 3D compartments within the real world using AR techniques; and third, biologic realistic mechanisms can be simulated by the ANN “GENESIS” and “NEURON”. Here, a common base for business process optimizations on behalf of neuronal mechanisms can be established, which includes the base for process optimization concepts considered for ANN learning tasks, and the tools identified are attractive for the CoNM foundation.

6.2 Neuronal Knowledge Modeling Language

The following demonstrates the CoNM meta-model and presents a CoNM process modeling language called *NMDL*, which has been designed and technically specified in section 4.6. Considering the relevant perspectives, it deals with the modeling of processes, business processes, knowledge processes, simulation processes, learning processes, training processes, testing processes, etc. The NMDL builds on modeling attempts provided by Grum and Gronau (2018a).

6.2.1 NMDL Modeling Neuronal Conversions

According to the four different kinds of conversions among knowledge bearers, which refer to the internalization, socialization, externalization and combination (cf. section 2.2.1.3), the following extends them by a neuronal interpretation. So, the very atomic conversions are visualized in Fig. 6.6.

As it was specified in section 4.6, the models clarify that an atomic neuronal conversion considers only one input object and only one output object. As compared to common modeling principles of the KMDL v.2.2, the knowledge object is designed to evolve because of a neuronal activity. This is why any tacit knowledge object is a successor of a neuronal activity. Further, the CoNM assigns a pattern of values to knowledge objects so that the signal flow throughout the ANN is objectified and can be visualized by the NMDL.

According to common modeling principles of the KMDL v.2.2, the complex neuronal conversion resembles the modeling of a complex conversion. Separating complex conversions in pure and impure conversions, Fig. 6.7 shows the four pure complex neuronal conversions summarizing atomic conversions of the same conversion type and only one neuron or group of neurons.

In the examples presented, each input object principally can lead to three individual output objects. Since each example shows three individual input objects, principally, nine kinds of atomic conversions have been summarized by each example. The concrete atomic neuronal conversion cannot be reproduced by this view only.

In the following three cases, one cannot find colored arrows linking objects: (1) Since *impure* complex neuronal conversions consider several neuronal conversion types, such that one is not able to distinguish the concrete underlying atomic neuronal conversion, here, the linking arrows have been visualized in black. (2) When the abstraction level is increased, and more than one transferring neuron or group of neurons participate in a conversion, this does not exclude the repetitive conversation of the same neuron so that one is not able to identify individual atomic knowledge flows of participating neurons, the linking arrows have been visualized in black. (3) Decreasing the granularity aggregates atomic neuronal conversions, and more than one transferring neuron or group of neurons participates at a conversion. This does not exclude the repetitive conversation of the same neuron; the linking arrows are visualized in black, too. All the three cases lead to the same visualization, which is presented in Fig. 6.8.

Here, one can see different kinds of knowledge objects, namely, tacit knowledge objects (e.g. see tacit knowledge object called *Tacit knowledge object A1.I*)

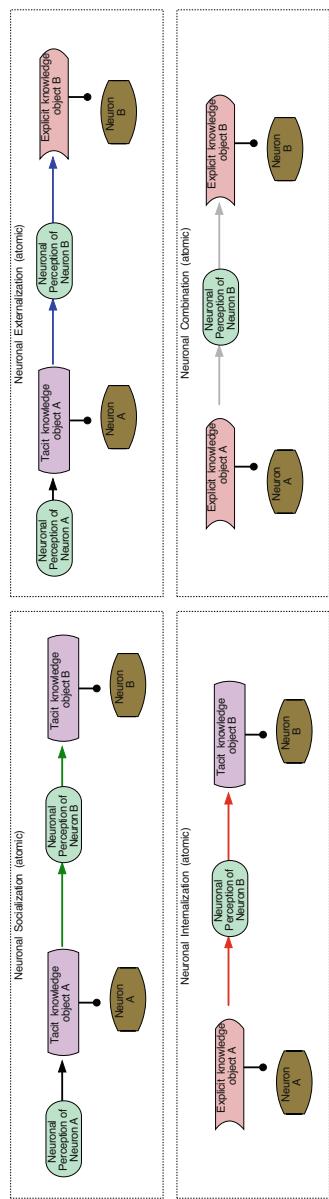


Figure 6.6 The Atomic Neuronal Conversions

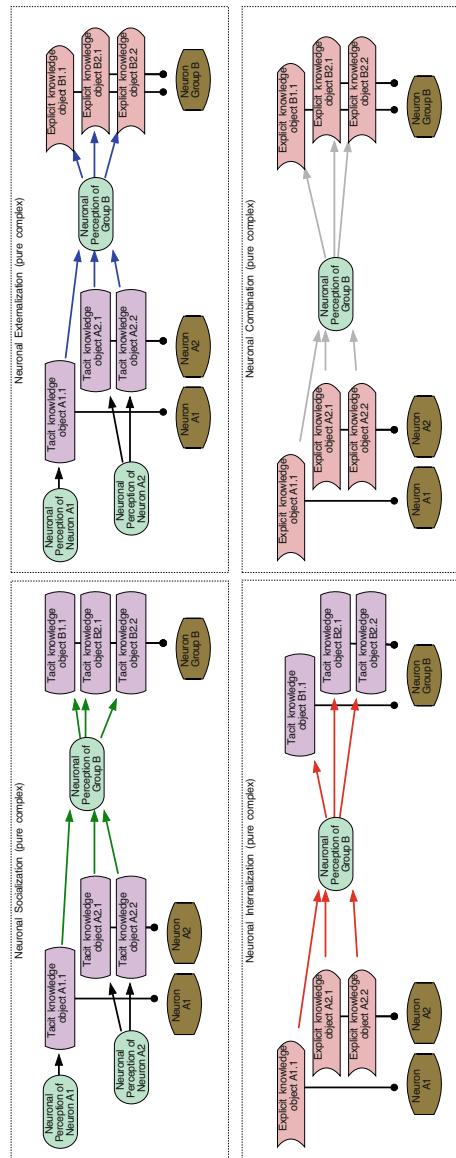


Figure 6.7 The Pure Complex Neuronal Conversions

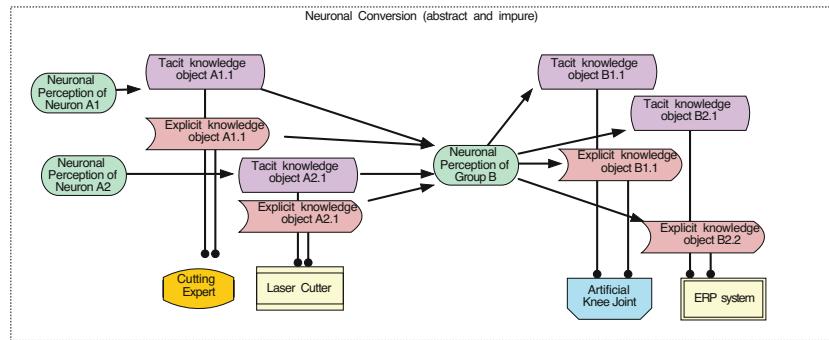


Figure 6.8 The Abstract Neuronal Conversions

and explicit knowledge objects (e.g. see explicit knowledge object called *Explicit knowledge object A1.1*).

Since these kinds of knowledge objects can either represent a detailed level or an abstract level, and these either can refer to an individual neuron (e.g. see activation called *Neuronal Perception of Neuron A1*), or a group of neurons (e.g. see activation called *Neuronal Perception of Group B*), an abstract labeling of knowledge objects has been chosen.

Although the CoNM assumes to approximate any kind of real world object and knowledge bearer by neurons, the abstract and impure complex conversions allow to visualize KM relevant entities in accordance with the NKM designed (cf. section 4.5). Hence, the knowledge objects are associated with neurons at the neuronal levels and represent information systems (e.g. see *ERP System*), tangible resources (e.g. see *Artificial Knee Joint*), machines (e.g. see *Laser Cutter*) or people (e.g. see *Cutting Expert*).

6.2.2 NMDL Modeling ANN Architectures

The following exemplifies the NMDL by the modeling of common network architectures, as they have been presented in section 2.5.5 and 2.5.8. As all the following cases are stored by the CoNM modeling repository (cf. NEA, section 4.3), each case can be reused for a quick start in neuronal modeling. In accordance with Def. 20, these can be interpreted as reference models of neural modeling. Even more sophisti-

cated reference models can represent e.g. machines or any form of processes (Def. 3) and BPs (Def. 4), as well as simulations (Def. 16) and knowledge transfers (Def. 14).

6.2.2.1 Case of Single Neurons

The individual neuron represents the most basic entity of an ANN. Via the composition edge, an axon hill can be explicitly associated with its neuron. This is required either for the biologic connection of dendrites and axons, or for the specification of the neuron activation function. Fig. 6.9 presents the kind of activation functions that are currently supported by the CoNM: the *bias* function that provides a constant signal of 1.0, the *linear* function that forwards input and error signals without any kind of squashing, and the common *TanH* and *Sigmoid* function that squashes signals.

Further, activations can be associated with the neurons, which allows one to take a deeper look at its processual meaning in the sense of actual entities.

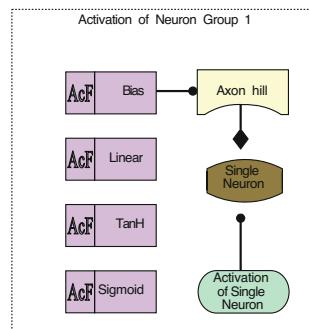


Figure 6.9 The Architecture of a Single Neuron (NMDL *NeuronView*)

6.2.2.2 Case of Multilayer Perceptrons

Multilayer Perceptrons (MLP) refers to a feedforward network, which contains various layers of at least one single neuron and provides directed connections from preceding neurons to consecutive neurons. Fig. 6.10 provides an example for a Boolean operator network having two inputs and one output.

Since linear layers are initialized per default for input and output neurons, neuron-specific axon hills and activation function items can be left out. Per default, any

kind of hidden layer neuron is initialized with the *Sigmoid* activation function. This supports the comfortable construction of clear models.

Since activations can be associated with individual neurons, a detailed look at its processual meaning within the joint ANN is enabled. This addresses its function in the sense of actual entities. Fig. 6.10 shows the association of the activation called *Activation of Hidden Neuron 4*.

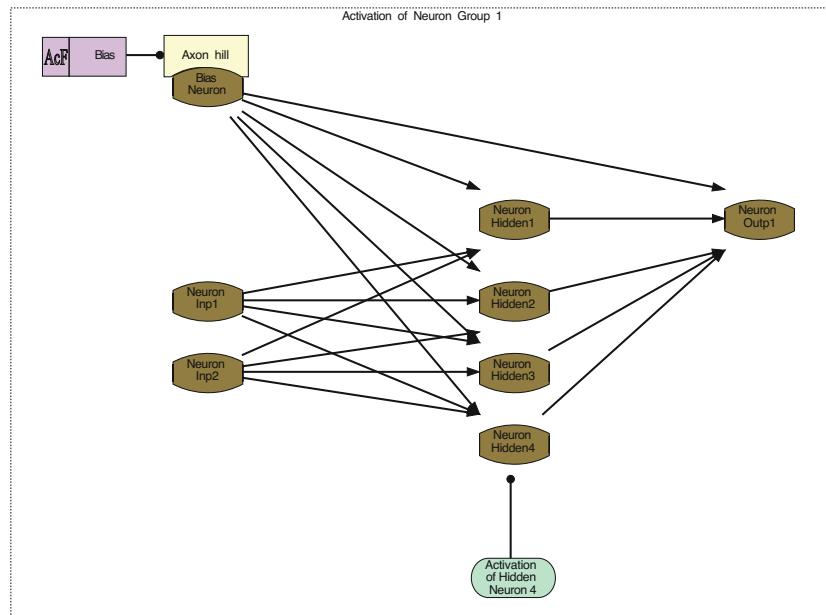


Figure 6.10 The Architecture of a MLP (NMDL *NeuronView*)

6.2.2.3 Case of Recurrent Neuronal Networks

If recurrent connections can be identified in the ANN, the network is referred to as recurrent neuronal network. Examples for recurrent connections can be found in Fig. 6.11 in all hidden layer neurons. These have been recurrently connected with themselves.

Since PyBrain does not support the recognition of feedforward or recurrent connections, the CoNM of the first version just supports recurrent connections of the

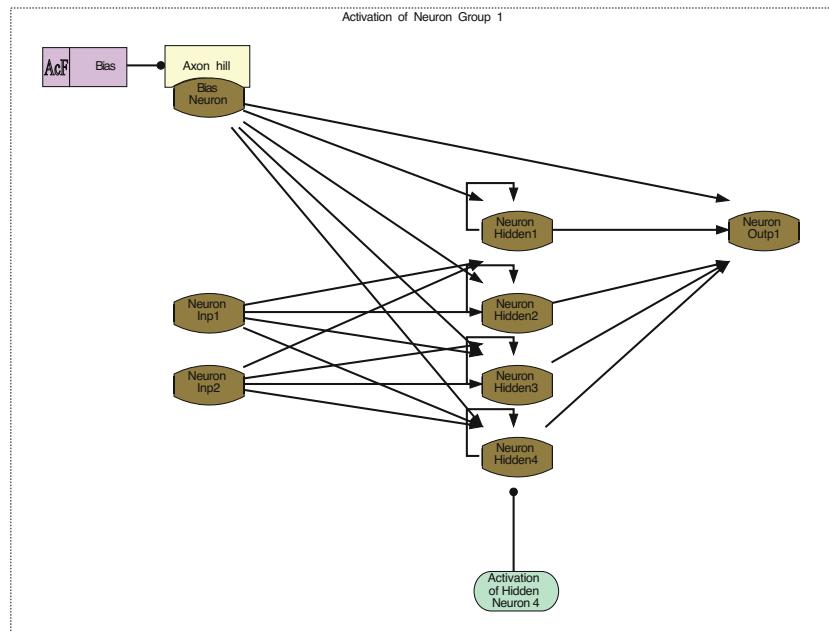


Figure 6.11 The Architecture of a RNN (NMDL *NeuronView*)

neuron itself. Later versions will need to recognize loops within networks so that the recurrence can be identified automatically.

Equivalent to the feedforward example presented in section 6.2.2.2 (Fig. 6.10), Fig. 6.11 shows the association of the activation called *Activation of Hidden Neuron 4*.

6.2.2.4 Case of Long-Short-Term-Memory Blocks

Fig. 6.12 presents the LSTM block that was specified with the aid of the NMDL *NeuronView*. Both, the LSTM block gates as well as their cells have been visualized by neurons. Please note here that the visualization does not show the technical connection of these neurons and the information flow. An example can be found at the filtering function of the forget gate on old cell signals. The NMDL visualization enables to focus on the interplay of its neurons with regard to knowledge generation.

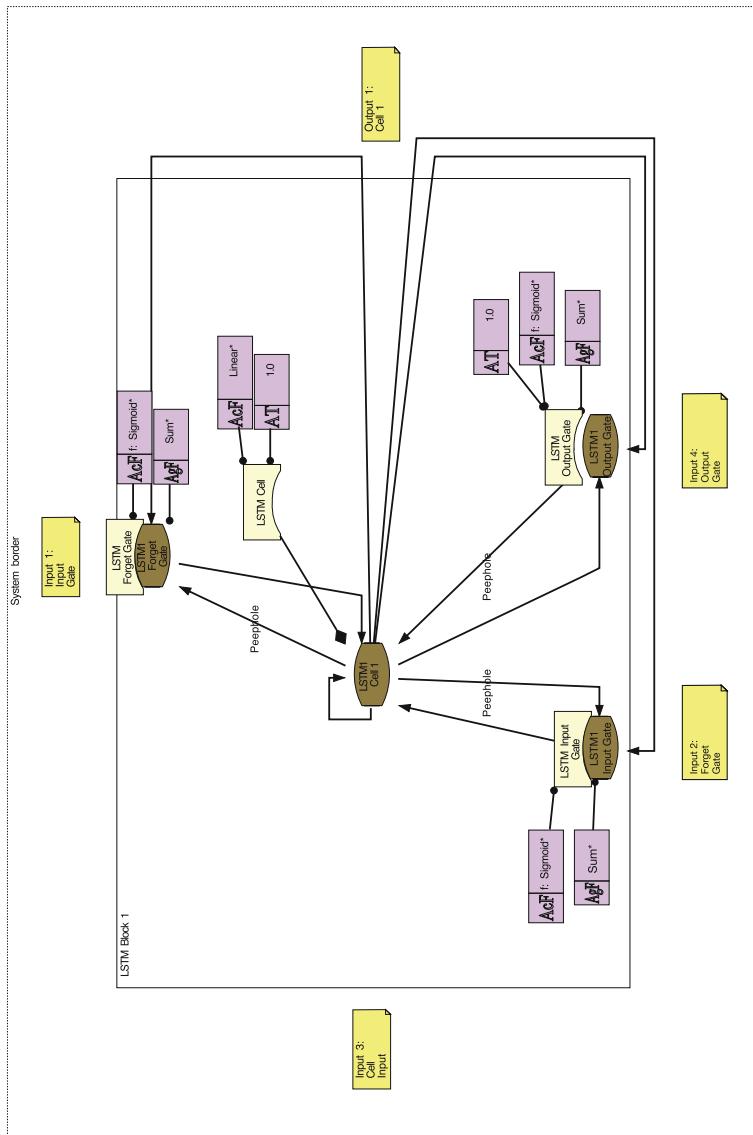


Figure 6.12 The Architecture of a LSTM Block (NMDL Neuron View)

Since PyBrain does not support LSTM blocks having more than one cell, the reuse of the LSTM block presented is recommended. Further, so far, the flexible use of peephole connections is not supported by PyBrain, whereas the CoNM enables the use of either all or none peephole connections. The same applies to the selection of the activation functions and aggregation functions. Thus, here as well as in all further figures, the same applies: Modeling items of the process models where the expressiveness of the CoNM exceeds the underlying programming library are marked by an *. A corresponding implementation would increase the potential of the underlying programming library. It would also be ideal for the CoNM.

6.3 Rise of Transparency

The following experiment intends to demonstrate the functioning of the CoNM with respect to the first sub-research question, that is, (*How can the modeling capability of knowledge modeling description languages be used to raise the transparency of NN?*), which can be found in section 1.2. A transparency in this context does not focus on the technical realization, for instance, on which device which blob or tensor is realized. It rather focuses on the operational performance of ANN so that changes in its structure and working behavior become clear.

6.3.1 Experiment 1—Live Learning

This experiment focuses on the demonstration of the visualization capabilities of the NMDL about the learning progress of an ANN. It enables cooperative modeling and process-oriented KM by both, human- and machine-based model creators, as it was designed using symbiotic KM approach in section 4.5. For instance, a process setting and ANN design is set up by a human model creator. This is complemented by the algorithmic training procedures of the ANN.

Problem Identification and Motivation: ANN can be considered tools for the arbitrary detailed system approximation. Since an approximation is based on learning procedures for ANN, which results because of the target-oriented adjustment because of thousands of parameters about the structural composition of an ANN, the visualization of structural parameters appears to be essential for the evaluation of the ANN's progress. Since the digital actor of an ANN has not been considered with regard to its processual intertwining within a network of processes so that it

manifests a behavioral process element, and its capability to generate knowledge so that it manifests as activity, one can identify a lack in the dealing with ANN.

By now, ANN have focused on the realization of predefined and isolated tasks. If it was possible to consider ANN to be tools for the approximation of any kind of process system and to discover based on a biologically plausible signaling how the performance of a system evolves, one can argue to operate on an activity level and process level.

Objective of the Solution: A modeling language was designed that visualizes any kind of process and ANN specification and modification so that a modeling space can function as a communication platform for all, humans as well as machine-based model creators. Further, a computational approach was designed (machine-based model creator) that visualizes any kind of structural change of an ANN in that modeling space in such a way that the inner composition such as neuronal modules, the connection among neurons, their direction, their weights as well as their inhibitory or excitatory function become transparent. These changes can help analyze the role of ANN as a knowledge bearer or rather as a process element.

A simple experiment setting was designed to demonstrate this, because even the smallest ANN provides thousands of elements and parameters. By this, a clear model construction as well as its straight-forward interpretation is supported, and the functioning of the following three becomes clear: First, the training specification, which includes testing; Second, the ANN learning behavior; Third, the ANN's processual intertwining.

Experimental Design: By following the CoNM proceeding (section 4.4), an ANN was set up having two input neurons, one bias neuron, four hidden layer neurons and one output neuron. Fig. 6.10 presents the NMDL *NeuronView* model of that structure as it was imported form the CoNM model repository (cf. NEA designed at Fig. 4.8) during the *design specification* phase (cf. CoNM proceeding designed at Fig. 4.9). Its detailed construction has already been presented in section 6.2.2.2 so that its composition of single neurons (Fig. 6.9) won't be issued in the following.

The training data holds sequences of one moment. These implicitly provide the XOR behavior. The test data provided the same kind of construction and can be seen in Fig. 6.13, which correspond to the NMDL *SetView* model and refers to a fixed learning task according to Def. 21.

In the figure, one can identify four sequences addressing an XOR case each. As the first and fourth sequence represent the XOR cases that need to produce 0.0, one can find an input of 0.0 at the neurons of the moment called *Moment border 1*, and one can find an input of 1.0 at the neurons of the moment called *Moment border*

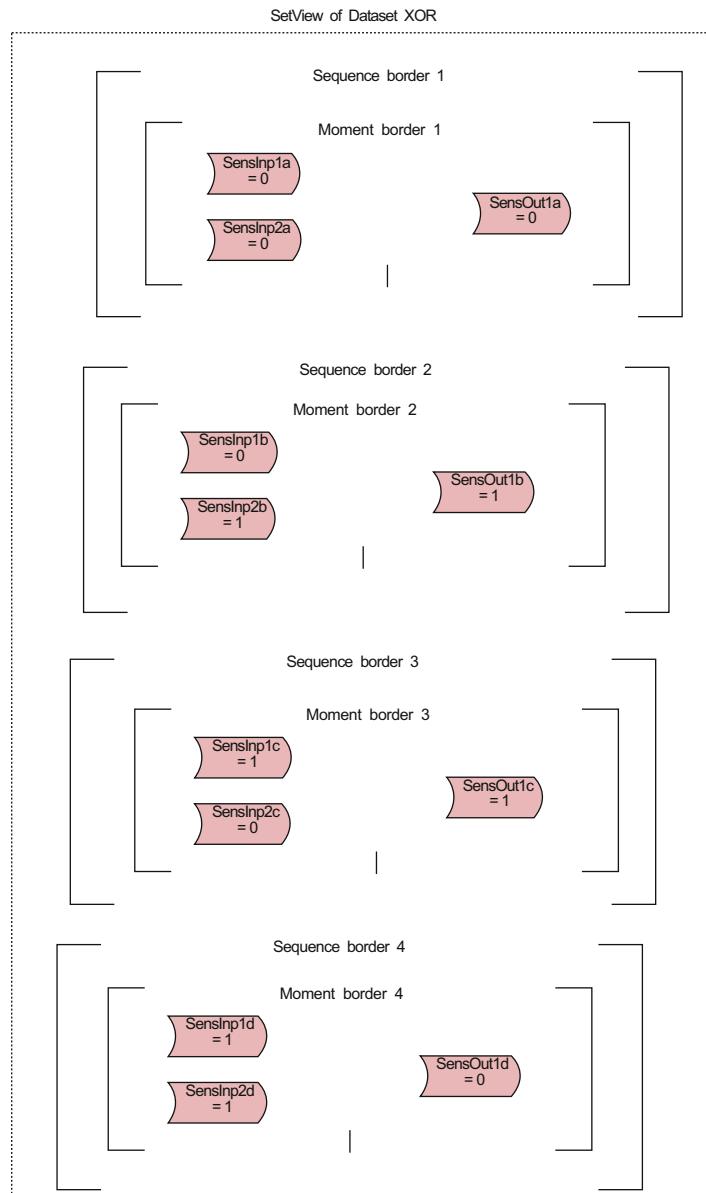


Figure 6.13 The Training Set for the Live Learning Demonstration (*SetView*)

4. The corresponding output refers to 0.0 at the moment called *Moment border 1* and the moment called *Moment border 4*. The second and third sequence represent the XOR cases that need to produce 1.0, which can be seen at the output of the moments called *Moment border 2* and *Moment border 3*. The corresponding inputs of the moments called *Moment border 2* and *Moment border 3* therefore refer to 0.0 and 1.0 or vice versa.

Please note that the naming of the explicit knowledge objects can show any kind of label that is plausible to the model creator, and the values can be found at the objects' attribute called *Knowledge Value*. This is essential for the corresponding knowledge composition. The corresponding NMDL *KnowledgeOverview* can be seen in Fig. 6.14.

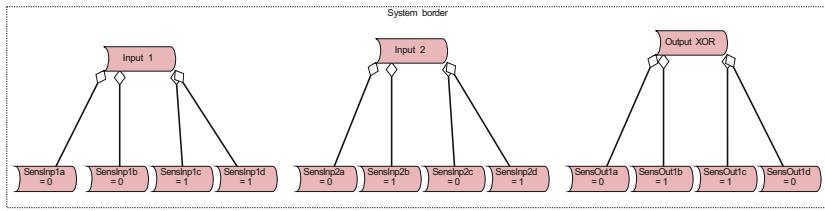


Figure 6.14 The Knowledge Composition for the Live Learning Demonstration (*KnowledgeOverview*)

As this figure shows the hierarchical composition of knowledge objects available and enables the from the viewpoint of the model creator , the interpretable and clear folding and unfolding of knowledge objects in any kind of view and model is prepared. This greatly supports the interpretation and dealing with knowledge.

As the actual entities will be realized based on the activations presented at the *NeuronView*, Fig. 6.15 presents the NMDL *NeuronalOverview*.

Here, one can see the composition of the group of neurons (NMDL *NeuronOverview*), the composition of the corresponding activations (NMDL *ActivationOverview*), as well as their association. Due to this, further views or models allow the folding or unfolding of activations corresponding to the underlying hierarchical decomposition. So, for example, the training can be specified on the aggregated level of the activation called *Activation of Neuron Group 1*. Because of the neuronal architecture specified (NMDL *NeuronView*), the atomic activation for individual neurons can be identified.

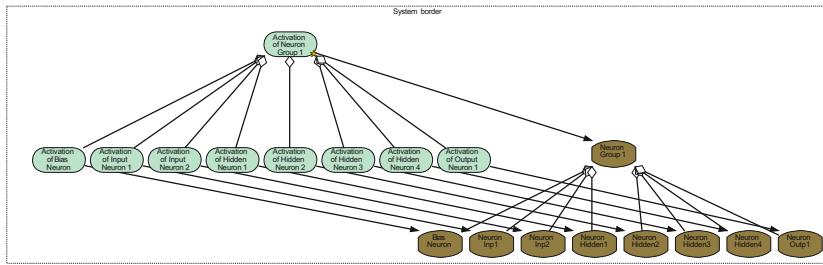


Figure 6.15 The Neuronal Composition of the ANN Trained (*NeuronalOverview*)

Training Phase: A learning task was specified by the NMDL *ActivationView* and can be found in Fig. 6.16.

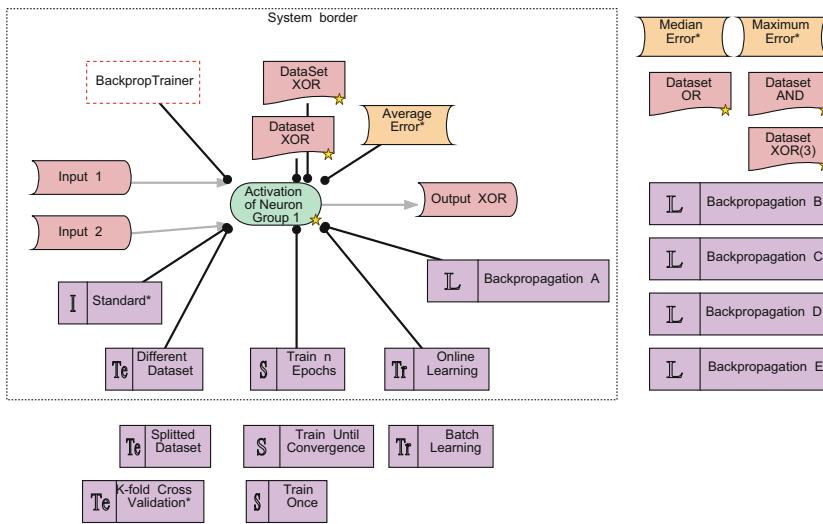


Figure 6.16 The Training Specification for the Live Learning Demonstration (*ActivationView*)

Here, one can see the activation of the corresponding networking structure called *Activation of Neuron Group 1*. This will be trained by the trainer called *Backprop-Trainer*. As its attribute called *Training Sessions* specifies, a training lasts for 1.000

sessions. Each session refers to one training run which has been specified by the stopping strategy item called *Train Once*. The trainer considers an *Online Learning* specified by the training strategy item and the learning strategy called *Backpropagation A*. By having a look at the learning strategy attributes, one can identify its specification of *learning rate*= 0.01, *momentum*= 0.99, *learning rate decay*= 1.0, *weight decay*= 0.0. Following the testing strategy item, an individual data set will be used for the training and testing. This refers to the data set called *Dataset XOR* that has been presented before. Its label was chosen with regard to the XOR case and was plausible for the human model creator. Alternative training configurations have been placed outside the system border so that these can be associated alternatively.

Testing Phase: The test data referred to the very same data set called *Dataset XOR*, as was indicated by the data set attribute called *Validation Dataset* in Fig. 6.16. According to the test strategy item, the joint data set was considered for this. Alternatively, by the choice of the *Splitted Dataset* test strategy, the data set could have been divided up into one part for the training and one part for the testing.

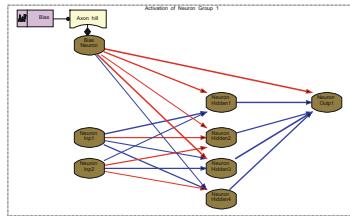
As the training was stopped after 1.000 training sessions, the desired quality was evaluated with just a slight deviation of the desired ANN performance. This referred to a deviation of not more than 0.1 from the desired output value. The calculation of the network's output has been visualized by Fig. 6.17 for a selection of learning steps.

In Fig. 6.17, in the first sub-figure column, one can see the ANN's performance of a certain learning step. Comparing these sub-figures, one can identify an improved performance by the learning steps, which is measured by individual errors, average errors, maxima errors and median errors. As Fig. 6.17(a) presents the performance of the initialization network, it shows the worst performance of all learning steps. The best performance with regard to all kinds of errors can be seen in Fig. 6.17(k).

The second sub-figure column presents the NMDL *NeuronView* of the ANN in that learning step so that the structural change of an ANN because of learning procedures becomes transparent. By comparing these sub-figures, one can identify the following: Fig. 6.17(b) presents a random initialization of the ANN having both, inhibitory and excitatory connections. Having realized 10 learning steps, Fig. 6.17(d) makes the effect of learning transparent and shows the change of the connection from the *Bias Neuron* to the neuron called *Neuron Hidden 3*. Initially, this had been excitatory (red color). Obviously, by looking at Fig. 6.17(f), because of the neuronal learning, the ANN confirms the tendency of negative input; all the connections from the input and bias layer to the hidden layer have changed and now are inhibitory. From the learning step 300 shown in Fig. 6.17(h), one can recognize the excitatory influence of the neuron called *Neuron Hidden4*. This is confirmed through learning

Seq.	Mom.	Input	Output	Correct	Error	
1	1	0,0	-1,524	0,0	-1,524	
2	1	0,0	-1,480	1,0	-2,480	
3	1	1,0	0,0	-0,986	1,0	-1,986
4	1	1,0	-0,824	0,0	-0,824	
Average Error:		1,6368879220546857				
Max Error:		3,0751548320232582				
Median Error:		1,9719352713809974				

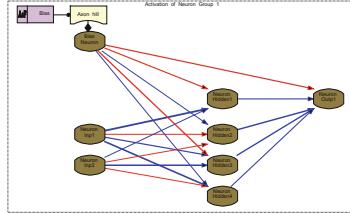
(a) ANN Performance at Step 0.



(b) ANN Structure at Step 0.

Seq.	Mom.	Input	Output	Correct	Error	
1	1	0,0	-1,057	0,0	-1,057	
2	1	0,0	-0,871	1,0	-1,871	
3	1	1,0	0,0	-0,672	1,0	-1,672
4	1	1,0	-0,484	0,0	-0,484	
Average Error:		0,95603095903644741				
Max Error:		1,7505004368500925				
Median Error:		1,3979172310104815				

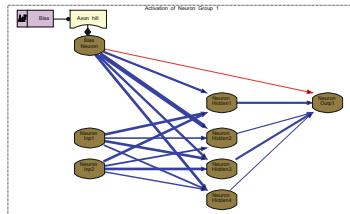
(c) ANN Performance at Step 10.



(d) ANN Structure at Step 10.

Seq.	Mom.	Input	Output	Correct	Error	
1	1	0,0	-0,063	0,0	-0,063	
2	1	0,0	0,349	1,0	-0,651	
3	1	1,0	0,0	0,346	1,0	-0,654
4	1	1,0	0,386	0,0	0,386	
Average Error:		0,12571068272653327				
Max Error:		0,21408249194332399				
Median Error:		0,21214629501569901				

(e) ANN Performance at Step 100.



(f) ANN Structure at Step 100.

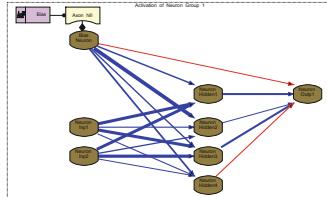
Figure 6.17 The Live Learning Visualization

as Fig. 6.17(j) shows. Now, the connection from the *Bias Neuron* via the hidden neuron called *Neuron Hidden4* to the output neuron called *Neuron Outp1* represents an excitatory connection. Hence, this kind of neuronal path seems to have a clear excitatory role within the ANN. This kind of ANN setup just changes a bit until the thousandth learning step (Fig. 6.17(l)). Apparently, faced by the clear performance improvement, a fine tuning is issued here.

Evaluation: The following evaluates to what extent the NMDL was able to raise transparency of NN. For this, the following will be based on the training specification as well as the course of learning visualization presented.

Seq.	Mom.	Input	Output	Correct	Error	
1	1	0,0	0,0	-0,050	0,0	-0,050
2	1	0,0	1,0	0,607	1,0	-0,393
3	1	1,0	0,0	0,570	1,0	-0,430
4	1	1,0	1,0	0,574	0,0	0,574
Average Error:		0,083916313568812767				
Max Error:		0,16482989839055934				
Median Error:		0,092274189240311347				

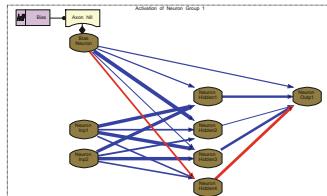
(g) ANN Performance at Step 300.



(h) ANN Structure at Step 300.

Seq.	Mom.	Input	Output	Correct	Error	
1	1	0,0	0,0	0,009	0,0	0,009
2	1	0,0	1,0	0,992	1,0	-0,008
3	1	1,0	0,0	0,986	1,0	-0,014
4	1	1,0	1,0	0,034	0,0	0,034
Average Error:		0,00018818139918083877				
Max Error:		0,00058920548433040682				
Median Error:		9,3086165871680468e-05				

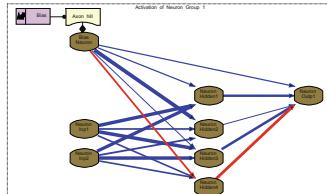
(i) ANN Performance at Step 500.



(j) ANN Structure at Step 500.

Seq.	Mom.	Input	Output	Correct	Error	
1	1	0,0	0,0	0,000	0,0	0,000
2	1	0,0	1,0	1,000	1,0	0,000
3	1	1,0	0,0	1,000	1,0	0,000
4	1	1,0	1,0	0,000	0,0	0,000
Average Error:		5,5847990258289772e-13				
Max Error:		1,6754610142822808e-12				
Median Error:		4,083905668652701e-13				

(k) ANN Performance at Step 1000.



(l) ANN Structure at Step 1000.

Figure 6.17 (continued)

Since the experiment demonstrated how to use the NMDL in order to specify a training and this specification was carried out by the CoNM mechanisms, the CoNM proved to successfully deal with ANN. Since not one line of programming code was needed for the training specification, one can identify an increased transparency after the set up of ANN and their training environment.

Further, the transparency was raised during the learning procedure. Since structural changes of the ANN have been visualized by NMDL models, the learning process became transparent. Particularly in terms of the characteristic to function exciting or inhibiting (edge color) and the influence of signals because of the weights

(edge thickness), the structural elements have been displayed transparently. Further tools, as they will be presented by later experiments, will even raise transparency, since they enable the model creators to intervene in the behavior of ANN.

On behalf of the storing of relevant parameters about the training, testing and the learning procedures, the human model creator has been enabled to recognize the current ANN performance during the course of learning. While the machine model creator carries out learning and modifies structural parameters of the ANN, the human model creator can intervene and modify learning based on stored models. For instance, the human model creator can stop learning procedures, change parameters manually, integrate further neurons or delete specific connections. Then, the human model creator can re-initiate learning procedures and continue the structural improvement.

Regarding the architectural setup of the CoNM implemented (cf. Fig. 5.3), which demands a central data base and a stream of data, changes within neuronal structures because of learning have been displayed live at corresponding devices. By this, a cooperative modeling of humans and machine model creators has been demonstrated. At any time, the human model creator was able to intervene and modify learning procedures. Assuming later CoNM versions to have a remote massaging system that cares about conflicting instructions, even the simultaneous modeling of several model creators within the AR or the 2D modeling space in *multiple-device-mode* has been prepared (cf. Fig. 5.1).

Communication: On behalf of the NMDL and the modeling environment of CoNM, relevant parameters for the training and testing of ANN as well as parameters affected by the learning procedures have been visualized. Further, they have been updated in a cooperative interplay of human and machine-based model creators so that a symbiotic modeling is enabled. Regarding the first research question, the experiment presented demonstrated an increase in transparency of ANN.

Contribution: In the context of the CoNM, the live learning functions as a tool whitening the black box of structural changes of an ANN training and testing. It further enables the cooperative modeling of human and machine-based model constructors.

6.4 Tacit Knowledge Identification

The following experiments intend to demonstrate the functioning of the CoNM with respect to the second sub-research question, that is, (*How can the learning*

capability of NN be used to identify tacit knowledge?), which can be found in section 1.2. Since the SEKO concept designed in section 4.7 provides different kinds of knowledge exhaustion approaches, the approaches have been demonstrated by individual experiments. The following will present each experiment in an individual subsection.

Although this research endeavors to demonstrate important views once, the detailed views, their technical linking via shortcuts and the navigation throughout all models can be seen by the corresponding CoNM project and its public repository.

6.4.1 Experiment 2—Static Knowledge Identification

This experimentation focuses on the demonstration of the SEKO approach that focuses on the static knowledge identification (SKI) of ANN and has been designed in section 4.7.1. Since these kinds of approaches lead to model elements that obviously cannot be found in reality, these have been treated as tacit knowledge and refer to an *extension mapping* (Fig. 2.1).

Problem Identification and Motivation: ANN can be considered tools for the arbitrary detailed system approximation. Being faced with great data sets and having numerous systems producing data, such as process participants, organizational resources, machines, etc., the behavior of any kind of system can be modeled by ANN in arbitrary detail.

Since the successful behavior of a participant cannot be explained by explicit knowledge objects only, tacit knowledge appears to be essential for that performance. Since explicit knowledge is the only kind of knowledge that can be externalized easily, one can identify a lack of identification of tacit knowledge.

By now, ANN have not been considered for the identification of tacit knowledge. If it was possible to consider ANN to be tools for the approximation of any kind of process system and to discover on the basis of a biologically plausible signaling how the performance of a system evolves, one can argue to identify tacit knowledge on a neuronal level.

Objective of the Solution: A computational approach was designed that approximates any kind of process resources as knowledge bearers, so that the inner working of its approximations can serve for analyses about tacit knowledge occurrence. For the exhaustion of knowledge, the SKI was issued.

A simple experiment setting was designed for this because even the smallest ANN provides thousands of elements and parameters. From this, a clear model construc-

tion as well as its straight-forward interpretation is supported, and the functioning of the SKI becomes clear.

Experimental Design: By following the CoNM proceeding (section 4.4), an ANN was set up having two input neurons, one bias neuron, one LSTM block in the hidden layer and one output neuron. Fig. 6.12 presents the NMDL *NeuronView* model of the LSTM block, as it was imported from the CoNM model repository during the *design specification* phase. Its detailed construction has already been presented in section 6.2.2.4. It was further surrounded by single neurons (Fig. 6.9), whose detailed construction has been presented in section 6.2.2.1.

The training data provided sequences of three moments. These implicitly provide the XOR behavior, which has a time delay of three time steps. The test data provided the same kind of construction and can be seen in Fig. 6.18, which corresponds to the NMDL *SetView* model. Please note that the sequences have been reorganized for the purpose of printing. Originally, the sequences were arranged vertically only.

In Fig. 6.18, one can identify four sequences addressing an XOR case each. As the first and fourth sequence represent the XOR cases that need to produce 0.0, one can find an input of 0.0 at the neurons of the moment called *Moment border 1a*, and one can find an input of 1.0 at the neurons of the moment called *Moment border 4a*. The corresponding output thus refers to 0.0 at the moment called *Moment border 1c* and the moment called *Moment border 4c*. Hence, the training data provides a fixed learning task according to Def. 21.

The second and third sequence represent the XOR cases that need to produce 1.0, which can be seen at the output of the moments called *Moment border 2c* and *Moment border 3c*. The corresponding inputs of the moments called *Moment border 2a* and *Moment border 3a* thus refer to 0.0 and 1.0 or vice versa.

As the system was designed to be based on impulses only, further values refer to 0.0. Please note that the naming of the explicit knowledge objects can show any kind of label, which must be plausible to the model creator, and the values can be found at the objects attribute called *Knowledge Value*. This is essential for the corresponding knowledge composition. The corresponding NMDL *KnowledgeOverview* can be found in Fig. 6.19.

Since this figure shows the hierarchical composition of the knowledge objects available and the NMDL allows the labeling of knowledge objects by the individual wordings of the model creator, the interpretable and clear folding and unfolding of knowledge objects in any kind of view and model is prepared for the modeling tool. The folding greatly supports the dealing with knowledge since the construction of clear models is supported. The unfolding, on the other hand, allows the display of

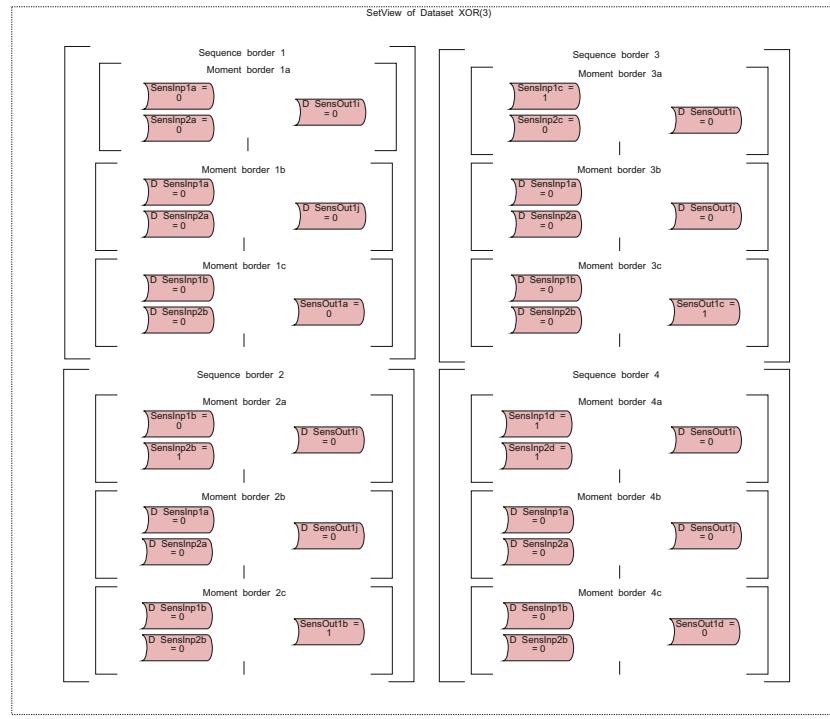


Figure 6.18 The Training Set for the SKI Demonstration (*SetView*)

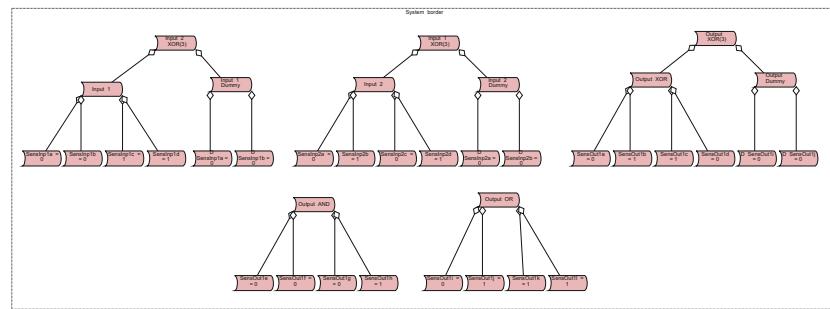


Figure 6.19 The Knowledge Composition for the SKI Demonstration (*KnowledgeOverview*)

more atomic kinds of knowledge objects so that the technical processing can be addressed.

As the SEKO will be realized based on the activations presented at the *NeuronView*, Fig. 6.20 presents the NMDL *NeuronalOverview*.

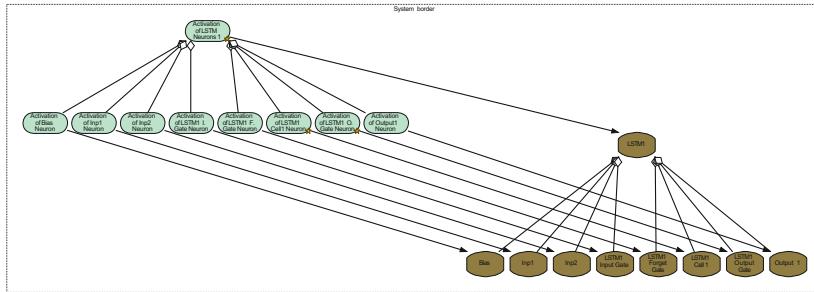


Figure 6.20 The Neuronal Composition of the ANN Trained (*NeuronalOverview*)

Here, one can see the composition of the individual neurons (NMDL *NeuronOverview*), the composition of the corresponding activations (NMDL *ActivationOverview*) and their association. Due to this, any view of a model allows the folding or unfolding of the activations corresponding to its hierarchical decomposition. So, for example, the training can be specified on the aggregated level of the activation called *Activation of LSTM Neurons 1*, and the neuronal architecture (NMDL *NeuronView*) can clarify the individual activation per neuron.

Training Phase: A learning task was specified by the NMDL *ActivationView* and can be found in Fig. 6.21. Here, one can see the activation of the corresponding networking structure called *Activation of LSTM Neurons 1*. This will be trained for one session by the trainer called *BackpropTrainer*. This session refers to 1.000 training runs, as has been specified by the attributes of the stopping strategy item called *Train n Epochs*. The trainer considers an *Online Learning* specified by the training strategy item and the learning strategy called *Backpropagation A*. Facing the detailed models, one can identify its attributes of *learning rate*= 0.01, *momentum*= 0.99, *learning rate decay*= 1.0 and *weight decay*= 0.0. Following the testing strategy item, an individual data set will be used for the training and testing. This refers to the data set called *Dataset XOR(3)*, which has been presented before. Its label was chosen with regard to the XOR having a time delay of three time steps.

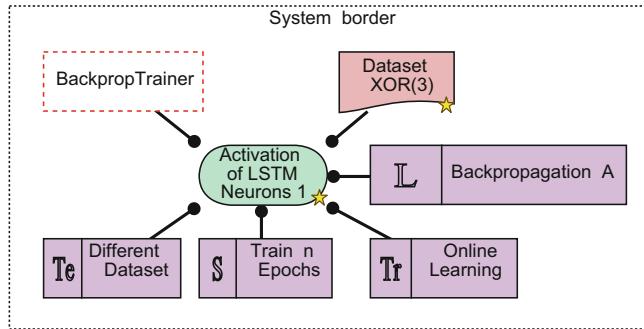


Figure 6.21 The Training Specification for the SKI Demonstration (*ActivationView*)

Testing Phase: The test data referred to the very same data set called *Dataset XOR(3)*, as was indicated by the data set attribute called *Validation Dataset* in Fig. 6.21. According to the test strategy item, the joint data set was considered for this. Alternatively, by the choice of the *Splitted Dataset* test strategy, the data set could have been divided into one part for the training and one part for the testing.

As the training was stopped after 1.000 training runs, the desired quality was evaluated with just a slight deviation of the desired ANN performance. This referred to a deviation of not more than 0.1 from the desired output value. The calculation of the network's output has been visualized by Fig. 6.22.

As the maximum error does not exceed the level of 0.0003 for the joint sequences and the individual errors per moment do not exceed the level of $0.038 < 0.1$, the trained network was chosen for completion by NPM, which, here, refers to the SEKO concept called *Static Knowledge Identification*. Fig. 6.23 presents the corresponding NPM output.

Here, one can see the LSTM network that was already specified during the training phase. Corresponding connections have been adjusted by the CoNM training procedures such that the edge's thickness corresponds to the absolute weight and the edge's color corresponds to the inhibition (negative weight) or excitation (positive weight). Further, corresponding to the neuron's displayed activation items, knowledge objects have been complemented by the NPM so that neuronal conversions can be derived on a neuronal level.

Seq.	Mom.	Input	Output	Correct	Error
1	1	0,0	0,0	0,000	0,000
	2	0,0	0,0	0,001	0,001
	3	0,0	0,0	0,011	0,011
2	1	0,0	1,0	0,001	0,001
	2	0,0	0,0	0,037	0,037
	3	0,0	0,0	1,004	0,004
3	1	1,0	0,0	0,001	0,001
	2	0,0	0,0	0,036	0,036
	3	0,0	0,0	0,983	-0,017
4	1	1,0	1,0	0,000	0,000
	2	0,0	0,0	0,000	0,000
	3	0,0	0,0	0,000	0,000
Average Error:		0,0001261054262635117			
Max Error:		0,00025984577921615897			
Median Error:		0,00022511304381360689			

Figure 6.22 The Performance of the ANN Trained

The neuron-specific neuronal arousal threshold is attached to the corresponding axon hill. For the SKI, only arousal threshold items are considered that contain the wording *Structural*. Again, further labeling can correspond to the model creator's preferences such that the interpretation is supported. Relevant values are specified by the item's attribute called *Arousal Threshold*, which have been identified manually in terms of the plausibility of NPM. If an activation does not provide an individual arousal threshold item, the corresponding attributes are set to the standard value of 1.0.

In accordance with the NEA presented in section 4.3, the models that have been constructed by the NPM are stored in the *Model Warehouse*. Using this, the knowledge identified and ANN structures constructed can be reused and adapted to other CoNM projects.

Evaluation: The following evaluates to what extent the knowledge derived is plausible. For this, the activations are considered layer-wise, starting at the back. While the following intends to present an overview, detailed labeling and individual modeling items can be found at the corresponding model of the public CoNM repository.

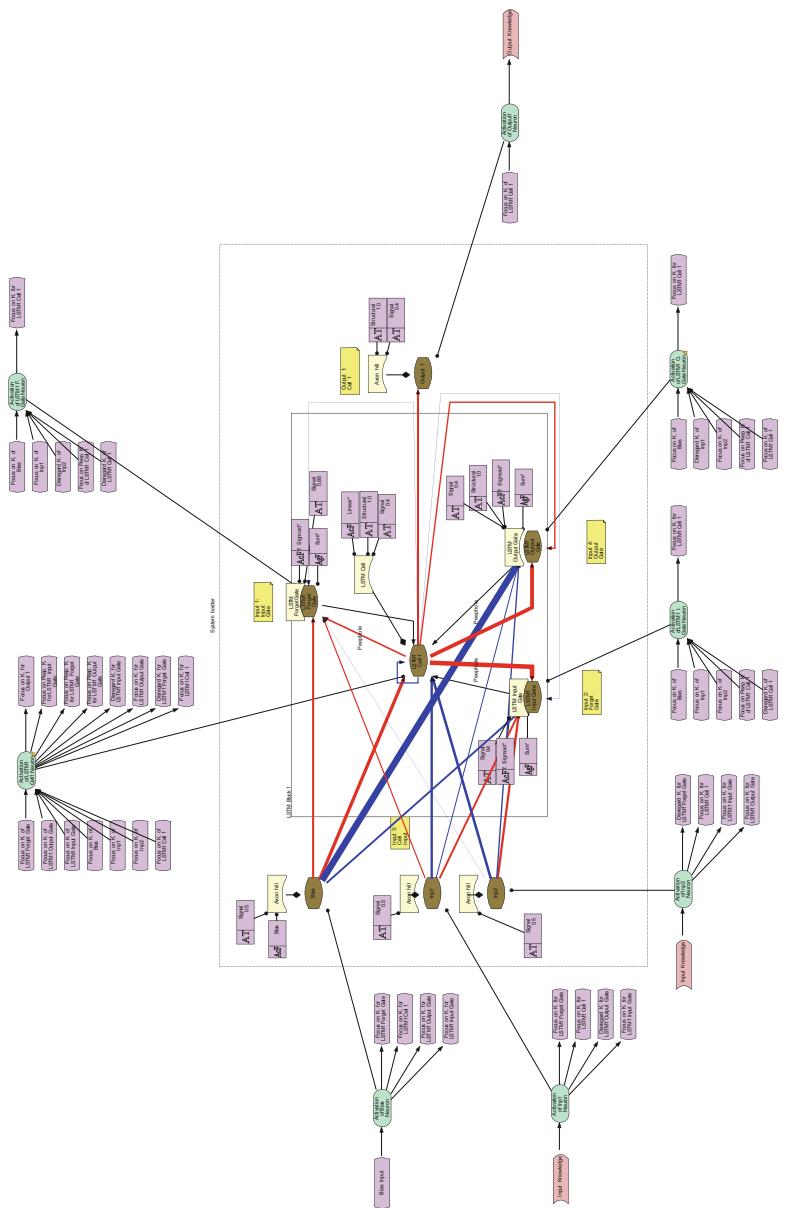


Figure 6.23 The SKI Demonstration (*Neuron View*)

1. *Activation of Output1 Neuron:* As the output neuron is only connected to the LSTM output, which is called *LSTM1 Cell1* here, it becomes clear that the neuron must focus on any kind of signal sent by its input neurons. Hence, a tacit knowledge object $+kb$ is identified (called *Focus on Knowledge of LSTM1 Cell1*), which is externalized so that the explicit knowledge object called *Output knowledge* can be identified.
2. *Activation of LSTM1 Output Gate Neuron:* How much of the current cell activity will be released is determined by the LSTM output gate. Here, structural knowledge objects clarify as to what kind of signals are relevant for the opening of the output gate. Since these objects are relevant, the objects are $+kb$ and are labeled as *Focus on Knowledge for LSTM1 Cell1*. Apparently, in this network, the opening is not really dependent on the signals coming from the input neurons. This becomes clear by the $-kb$ labeled as *Disregard Knowledge of Inp1* and *Disregard Knowledge of Inp2*. The focus lies on the signals coming from the bias neuron (see *Focus on K. of Bias*) as well as both kinds of recursive connections coming from the LSTM block, namely, the peephole connection (see *Focus on Peep. K. of LSTM1 Cell1*) and the recurrent connection from the LSTM block (see *Focus on K. of LSTM1 Cell1*). Hence, the output gate opening is dependent on the long-term memory of the LSTM block. As the bias neuron inhibits the output gate strongly (blue colored connections), aside from the inhibiting influence of input neurons, the output gate is forced to be closed for as long as the exciting signals of the *LSTM1 Cell1* compensate the inhibitory influence (red colored connections).
3. *Activation of LSTM1 Forget Gate Neuron:* How much of the current cell activity will be forgotten is determined by the LSTM forget gate. Here, structural knowledge objects clarify which kind of signals are relevant for the opening of the forget gate. These objects indeed are relevant, because of which they are called *Focus on Knowledge for LSTM1 Cell1*. Apparently, in this network, the degree of forgetting is neither dependent on the signals coming from the neuron called *Inp2* nor (see *Disregard K. of Inp2*) on the signals coming from the recurrent connection of the LSTM block (see *Disregard K. of LSTM1 Cell1*). This becomes clear by the $-kb$, which is called *Disregard Knowledge of Inp2* and *Disregard Knowledge of LSTM1 Cell1*. The neurons focus knowledge $+kb$ lays on signals coming from the bias neuron called *Inp1* and the neuron called *LSTM1 Cell1* via the peephole connections (see *Focus on Peep. K. of LSTM1 Cell1*). In accordance with the connection colors, the forget gate seems to be intending to make the LSTM cell remember the current activation accumulated over the course of time.

4. *Activation of LSTM1 Cell1 Neuron:* Being situated right within the center of the ANN, the neuron called *LSTM1 Cell1* considers all input knowledge objects to be focus objects $+kb$. All output objects are considered to be $+kb$ with two exceptions: Obviously, only signals being transmitted via the recurrent connections to the input gate and to the forget gate can be identified as $-kb$. Regarding the connection colors, the cell seems to excite surrounding neurons except itself. This results from the continuous exciting signals coming from the bias neuron, which are compensated for by the inhibiting signals from the input neurons and the cell itself.
5. *Activation of LSTM1 Input Gate Neuron:* How much of the current cell activity will be enriched by incoming signals is determined by the LSTM input gate. Here, the structural knowledge objects clarify as to which kind of signals are relevant for the opening of the input gate. These objects are relevant for the LSTM block; so, they are called *Focus on Knowledge for LSTM1 Cell1*. Apparently, in this network, the degree of input is not dependent on the signals coming from the recurrent connection of the LSTM block. This becomes clear by the $-kb$ called *Disregard Knowledge of LSTM1 Cell1*. The neurons focus knowledge $+kb$ lays on signals coming from the bias neuron, the input neurons called *Inp1* and *Inp2* and the neuron called *LSTM1 Cell1* via the peephole connections (see *Focus on Peep. K. of LSTM1 Cell1*). Regarding the connection colors, it seems that the inhibiting influence of the bias neuron signals are compensated for by the exciting input neuron signals. Hence, as soon as an activation is sent from the input neurons, it is allowed to enter the LSTM cell.
6. *Activation of Bias Neuron:* The neuron called *Bias* consequently considers the knowledge called *Bias Input*. As the output objects identified state it clearly, any kind of connection is relevant. Therefore, any kind of knowledge identified is declared as $+kb$.
7. *Activation of Input 1 Neuron:* As the input neuron *Inp1* receives external signals, an explicit knowledge object $+kb$ is identified (called *Input Knowledge*). These are internalized so that the following tacit knowledge objects can be identified. Apparently, the signal transmission to the input gate, to the forget gate and to the cell are relevant so that the focus objects $+kb$ called *Focus Knowledge for LSTM1 Forget Gate*, *Focus Knowledge for LSTM1 Cell1* and *Focus Knowledge for LSTM1 Input Gate* can be identified. Signals being transmitted to the output gate seem to be irrelevant; so, the object $-kb$ called *Disregard Knowledge for LSTM1 Output Gate* can be identified.
8. *Activation of Input 2 Neuron:* As the input neuron *Inp2* receives external signals, an explicit knowledge object $+kb$ is identified (called *Input Knowledge*). These are internalized so that the following tacit knowledge objects can be identified.

Apparently, the signal transmission to the input gate, to the output gate and to the cell are relevant, so that the focus objects $+kb$ called *Focus Knowledge for LSTM1 Output Gate*, *Focus Knowledge for LSTM1 Cell* and *Focus Knowledge for LSTM1 Input Gate* can be identified. Signals being transmitted to the forget gate seem to be irrelevant; hence, the object $-kb$ called *Disregard Knowledge for LSTM1 Forget Gate* can be identified.

Regarding the knowledge objects identified, one can record that the behavior interpreted corresponds to the design intention of LSTM blocks (Hochreiter and Schmidhuber, 1997b).

Aside the visualization of knowledge objects identified by the NMDL *Neuron-View*, these objects can be visualized by the NMDL *KnowledgeGenerationView* (see Fig. 6.24). Using this, the knowledge flow throughout the activities and processes as well as neuronal conversions become clear, as they have been specified in section 6.2.1. So, the transparency of the ANN can be increased since the generation, transfer and application of knowledge becomes transparent. This particularly shows strengths in terms of knowledge flow among various knowledge bearers. The following presents an example limited to one ANN only.

Here, one can identify that the knowledge constructed strongly is based on the interplay of the ANN's neurons. Beside the explanations given before, this becomes transparent by the focus knowledge objects being given by all the activations. Since recurrent connections are not explicitly present here, they can be implicitly recognized by knowledge objects being associated with neurons of succeeding neurons. Fig. 6.25 presents a variation that presents recurrent conversions in terms of their output objects explicitly.

Communication: On behalf of the SKI, the interpretation of the ANN structure was enabled. Knowledge objects identified made the signaling focus transparent so that it became clear as to which kind of signals are relevant from the viewpoint of a neuron. This includes both, the relevance for the knowledge receiving and the relevance for the knowledge transmission. Particularly, in terms of their characteristic to function as exciting or inhibiting (edge color) and the influence of signals because of the weights (edge thickness), the interpretation of knowledge objects identified have been relativized plausibly.

This gives evidence that the NPM enables the plausible interpretation of ANN and increases the transparency in terms of the crystallized knowledge codified within the ANN (first research question). Further, the transparency of the ANN is increased as the knowledge generation, transfer and application is visualized by the *KnowledgeGenerationView*. Regarding the second research question, this gives evidence

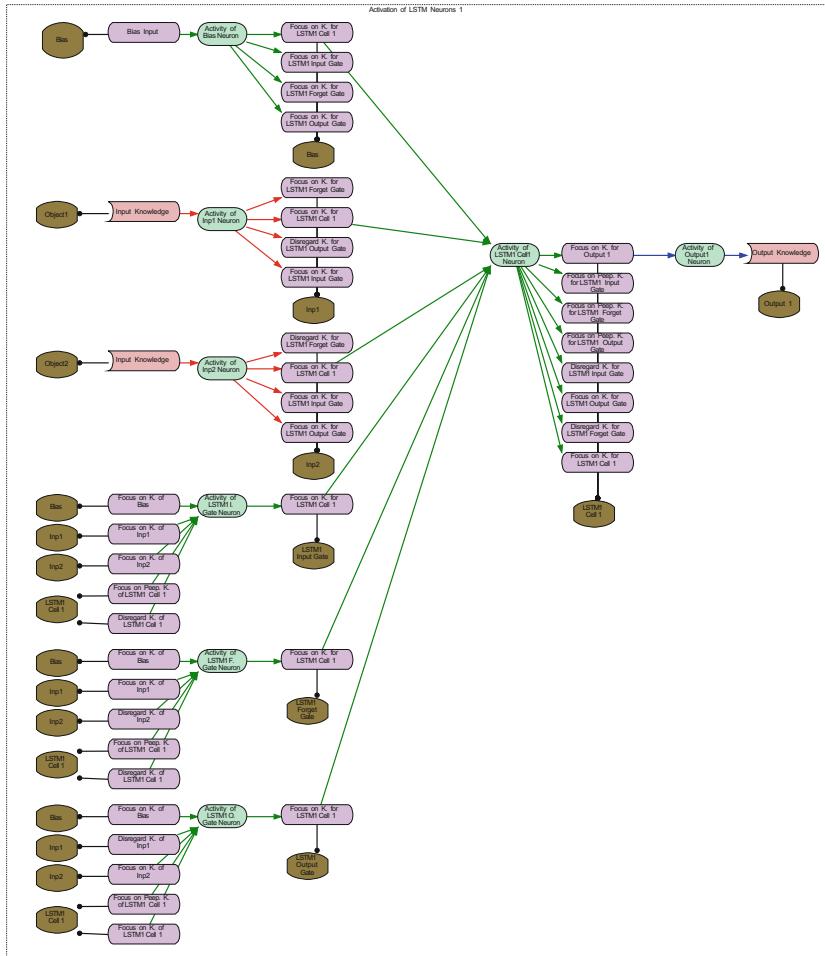


Figure 6.24 The SKI Demonstration (*KnowledgeGenerationView* Variant 1)

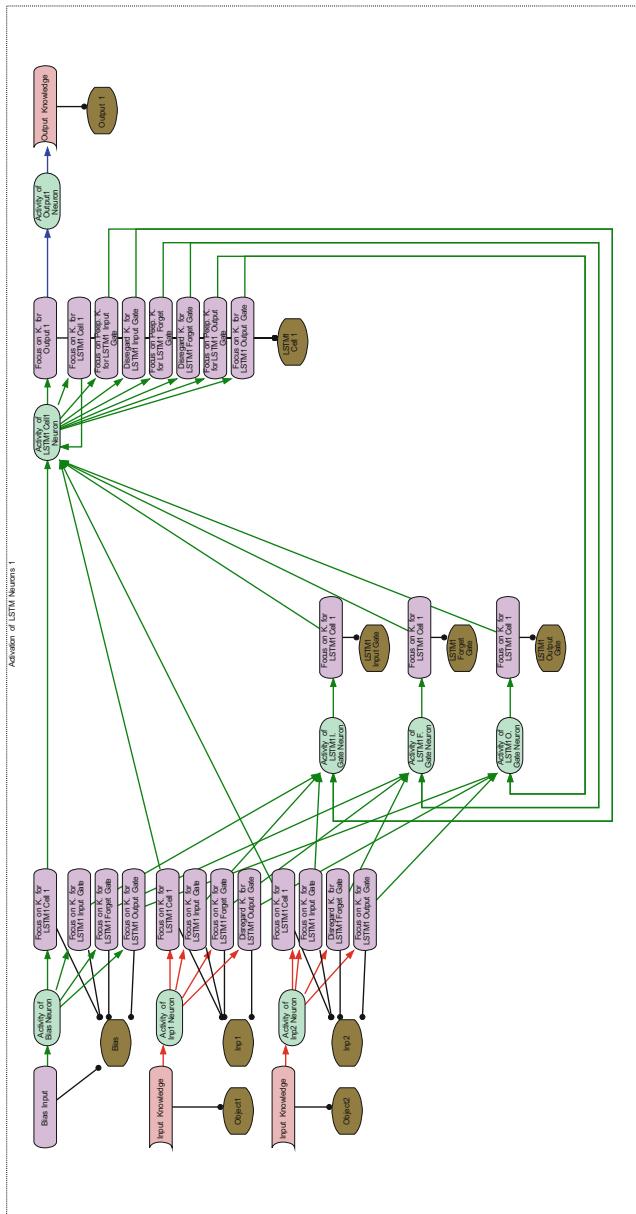


Figure 6.25 The SKI Demonstration (*KnowledgeGenerationView Variant 2*)

regarding the identification of tacit knowledge with the aid of the NMDL. Using this, neuronal conversions have been visualized in an experimental setting for the first time.

Contribution: In the context of the CoNM, the SKI functions as a tool whitening the black box of ANN using structural knowledge. It further enables the cooperative modeling of human and ANN-based model constructors, as knowledge objects can be identified by both. These will be the foundation for the process-oriented KM that symbiotically considers human as well as machine-based knowledge bases (cf. section 4.5).

6.4.2 Experiment 3—Dynamic Knowledge Identification

This experimentation focuses on the demonstration of the SEKO approach that focuses on the dynamic knowledge identification (DKI) of ANN and has been designed in section 4.7.1. As these lead to model elements that obviously cannot be found in reality, these have been treated as tacit knowledge and refer to an *extension mapping* (Fig. 2.1).

Problem Identification and Motivation: As this experiment focuses on a variation of the SEKO, the very same problem and motivation can be identified here, which has been presented in section 6.4.1.

Objective of the Solution: As compared to the SKI issued in section 6.4.1, a computational approach was designed that approximates any kind of process resources as knowledge bearers such that the inner working of its approximations can serve for analyses about tacit knowledge occurrence. For the exhaustion of knowledge, the DKI was issued.

As this approach refers to a SEKO variant, the very same experiment setting was chosen to demonstrate DKI than the SKI was demonstrated with. The simple experiment setting chosen supports the educative and a clear model construction as well as its straight-forward interpretation so that the functioning of the DKI becomes clear.

Experimental Design: The experimental design has considered the very same network architecture that has been presented in Fig. 6.23 and trained in section 6.4.1. During the *design specification* phase, it was imported from the CoNM model repository (cf. reference models, Def. 20).

The training data referred to the XOR behavior, which has a time lag of three time steps and can be seen in Fig. 6.18. Hence, it refers to a fixed learning task according to Def. 21.

Training Phase: Since the trained network shown in section 6.4.1 has been imported from the CoNM model repository, the learning task can be reproduced by Fig. 6.21 presented in the previous experiment.

Testing Phase: The test data referred to the data set called *Dataset XOR(3)*, which is essential for the DKI because the signals of each moment are considered individually. For the model completion using NPM, the trained network was chosen and the SEKO concept called *Dynamic Knowledge Identification* was applied. Fig. 6.26 presents the corresponding NPM output in accordance with all the sequences available in *Dataset XOR(3)*. Since the models focusing on individual moments and individual sequences can be found at the public CoNM repository, these maybe do not provide an overwhelming amount of knowledge; the following presents the joint knowledge base of all kinds of knowledge $+KB$ and $-KB$ only. Although the filtered models would appear clearer than the joint model, the focus was set on the visualization of the total knowledge base.

In the figure, one can see the LSTM network that has been specified and presented in Fig. 6.23. As here the trained network has been considered for the NPM, corresponding connections have been adjusted by the CoNM training procedures so that the edge's thickness corresponds to the absolute weight and the edge's color corresponds to the inhibition (negative weight) or excitation (positive weight). Further, corresponding to the neuron's displayed activation items, knowledge objects have been complemented by the NPM so that neuronal conversions can be derived on a neuronal level.

The neuron-specific arousal threshold is attached to the corresponding axon hill. For the DKI, only arousal threshold items are considered that contain the wording *Signal*. Again, the further labeling can correspond to the model creator's preferences so that the interpretation is supported. Relevant values are specified by the item's attribute called *Arousal Threshold*, which have been identified manually in regard with the plausibility of NPM. If an activation does not provide an individual arousal threshold item, the corresponding attributes are set to the standard value of 1.0.

In accordance with the NEA presented in section 4.3, models that have been constructed using the NPM are stored in the *Model Warehouse*. This shows that knowledge identified and ANN structures constructed can be reused and adapted to other CoNM projects.

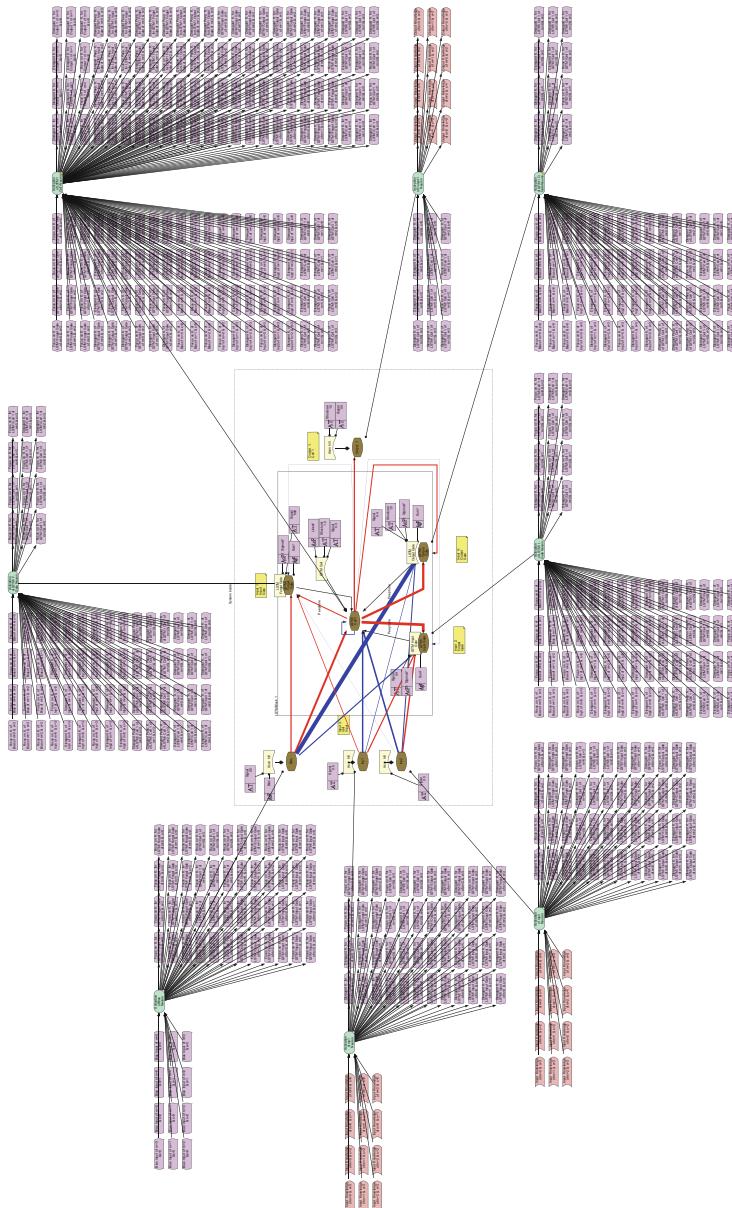


Figure 6.26 The DKI Demonstration (*NeuronView*)

Evaluation: The following evaluates to what extent the knowledge derived is plausible. For this, the activations are considered layer-wise, starting at the back.

1. *Activation of Output1 Neuron:* As the output neuron is only connected with the LSTM output, which is here called *LSTM1 Cell1*, it becomes clear that the neuron must consider any kind of signal sent via its input connections. Here, only the knowledge being transmitted by the third moments of the second and third sequence or rather scenario are relevant, which refer in the first case to an arousal of 0.44893006496 (*Focus on K. of LSTM1 Cell 1 of m=2 & s=1*) and in the second case to an arousal of 0.439540373565 (*Focus on K. of LSTM1 Cell 1 of m=2 & s=2*). Hence, a tacit knowledge object $+kb$ is identified in these cases that is externalized so that the explicit knowledge object called *Output Knowledge of m=2 & s=1* and *Output Knowledge of m=2 & s=2* can be identified. Further, input knowledge objects refer to tacit knowledge of $-kb$, all having values around 0.0. Output knowledge objects refer to explicit knowledge of $-kb$, and Fig. 6.22 presents an overview of the corresponding knowledge object attributes called *Arousal* at the column called *Output*.
2. *Activation of LSTM1 Output Gate Neuron:* For the opening of the output gate, one can see that the neuron focuses on the third moments of the second and third scenario (*Focus on K. for LSTM1 Cell 1 of m=2 & s=1* and *Focus on K. for LSTM1 Cell 1 of m=2 & s=2*). Obviously, the neuron has learned to unleash the currency collected at the *LSTM1 Cell1* exactly at that time step when the output is needed at the ANN's output. Other outgoing knowledge objects of the output gate are $-kb$ and refer to values around 0.0. The detailed labeling and values can be found at the corresponding model of the CoNM repository.
The input objects of the output gate activation focus on the bias signals at nearly all moments of all sequences, which seems to be inhibiting basic activation required for the consequent closing of the output gate. This becomes clear from the corresponding blue edge color. As the arousal at the *LSTM1 Cell1* indicates the start of a sequence from the perspective of the ANN, it relativizes negative signals coming from the bias neuron via the peephole connection (see *Focus on Peep K. of LSTM1 Cell 1 of m=2 & s=1* and *Focus on Peep K. of LSTM1 Cell 1 of m=2 & s=2*) as well as via the recurrent connection (see *Focus on K. of LSTM1 Cell 1 of m=2 & s=1* and *Focus on K. of LSTM1 Cell 1 of m=2 & s=2*) right at the moment to open the gate. Only in the case of the fourth scenario, which means having both the input neurons activated at the first moment, the inhibiting signal does not allow the opening of the output gate. This is indicated by the tacit knowledge objects called *Focus on K. of Inp1 of m=0 & s=3* and *Focus on K. of Inp2 of m=0 & s=3*. Apparently, at least one activation is required for the

opening of the gate, which is in scenario 2 the object called *Focus on K. of Inp2 of m=0 & s=1* and in scenario 3 the object called *Focus on K. of Inp1 of m=0 & s=2*. With this, all the cases the output gate must deal with can be explained plausibly.

3. *Activation of LSTM1 Forget Gate Neuron:* For the opening of the forget gate, which means remembering, one can identify that the neurons focus on the moments of nearly all sequences. Here, they provide *Arousal*s greater than 0.88 so that signals of the cell are remembered to a greater extent. The fourth scenario seems to be an exception, as the signals of the second and third moments are not remembered to that degree. Obviously, here, the input signals from the bias neuron or one of the input neurons is missing.

Regarding the input objects, which cause knowledge generation at the forget gate neuron described, beside the bias signals of all moments, input signals are focused as $+kb$ when they refer to the level of 1.0. Any kind of different signals are considered to be $-kb$.

4. *Activation of LSTM1 Cell1 Neuron:* The LSTM cell neuron called *LSTM1 Cell1* cares about the long-term storing of the ANN's activation. Apparently, this refers to the collection of an adequate high arousal, which is required right in that moment the intended ANN output is to be produced. This becomes clear by the provision of the focus knowledge $+kb$ at the third moment of the second and third scenario, which is relevant for any kind of connection to consecutive neurons. These have been labeled by the NPM algorithm as *Focus on K. for Output 1 of m=2 & s=1* and *Focus on K. for Output 1 of m=2 & s=2* in regard with the output neuron. Plausibly, knowledge of any other kind of moments and scenarios, or rather sequences, refer to $-kb$ and is about the disregarding of corresponding signals.

Regarding the input objects, which cause the knowledge generation at the LSTM cell neuron described, the following $+kb$ knowledge is required in order to realize a complex interplay: First, apparently signals coming from the forget gate are focused at all moments and sequences so that a continuous storing is realized. Second, signals coming from the bias neuron are focused at all moments and sequences, which seem to provide a kind of basic activation. Third, only at the second and third scenario the knowledge from the output gate is focused, which is responsible for the opening of the output gate right in that moment when the current cell activation needs to be unleashed in order to produce the intended ANN output. This can be identified by the tacit knowledge objects $+kb$ called *Focus on K. of LSTM1 Output Gate of m=2 & s=1* and *Focus on K. of LSTM1 Output Gate of m=2 & s=2*. Previous moments of sequence two and three as well as the other moments refer to knowledge of $-kb$ and consider disregarding

the cell's current activation. Fourth, the input gate focuses on all the moments of the second and third sequence. Apparently, this is essential for the consideration of relevant input activation signals called *Focus on K. of Inp2 of m=2 & s=1* and *Focus on K. of Inp1 of m=2 & s=2* as well as the consideration of recurrent signals coming from the LSTM cell called *Focus on K. of LSTM1 Cell 1 of m=2 & s=1* and *Focus on K. of LSTM1 Cell 1 of m=2 & s=2*. Apparently, in the fourth scenario, in which both input neurons are activated at the first moment (see *Focus on K. of Inp1 of m=0 & s=3* and *Focus on K. of Inp2 of m=0 & s=3*), the twofold input activation knowledge is used in order to disregard input gate knowledge. This becomes clear from the consideration of the input knowledge *Focus on K. of LSTM1 Input Gate of m=0 & s=3* at the first moment only. Other moments refer to the disregarding of the input gate knowledge $-kb$.

With this, all the cases the cell must deal with can be explained plausibly.

5. *Activation of LSTM1 Input Gate Neuron:* For the opening of the input gate, which means the selection of signals for them to be remembered by the *LSTM1 Cell1*, one can see that the cell seems to be protected for signals. Only in the second and third scenario, the gate opens drastically, which majorly makes the LSTM block consider bias input and recurrent cell input. This seems to be essential for the desired output to be generated. Although it is opened in the fourth scenario at the first moment as well (see *Focus on K. for LSTM1 Cell 1 of m=0 & s=3*), later moments do not consider bias signals and recurrent signals from the cell to a great extent.

Regarding the input objects, which cause knowledge generation at the input gate neuron described, beside bias signals of all moments, and the activation signals coming from the input neurons at the first moment, the second and third scenario focus on signals coming via the peepholes (see *Focus on Peep K. of LSTM1 Cell 1 of m=2 & s=1* and *Focus on Peep K. of LSTM1 Cell 1 of m=2 & s=2*) as well as via the recurrent connection (see *Focus on K. of LSTM1 Cell 1 of m=2 & s=1* and *Focus on K. of LSTM1 Cell 1 of m=2 & s=2*) right at the moment to produce the desired output activation. Hence, even the input gate seems to participate in the opening because it floods the cell by further activation. Other signals are considered to be knowledge about the disregarding $-kb$ and are essential for the generation of the desired ANN activation of 0.0.

6. *Activation of Bias Neuron:* The *Bias* neuron injects a kind of basic activation in the ANN. So, for all the moments and sequences, one can identify focus knowledge objects $+kb$ that are transmitted via the connections among the bias neuron and its consecutive neurons.

Regarding the input objects, one can therefore identify corresponding tacit knowledge objects $+kb$ during all moments and sequences.

7. *Activation of Input 1 Neuron:* For the activation of the network, the input neuron called *Inp1* provides the focus knowledge $+kb$ for the three kinds of LSTM gates and its gated LSTM cell at the first moment of the third and fourth scenario only. This refers to the relevant XOR activation, and corresponding knowledge objects are labeled by *Focus on K. for LSTM1 Forget Gate of $m=0 \& s=2$* and *Focus on K. for LSTM1 Forget Gate of $m=0 \& s=3$* for the forget gate, *Focus on K. for LSTM1 Cell1 of $m=0 \& s=2$* and *Focus on K. for LSTM1 Cell1 of $m=0 \& s=3$* for the LSTM cell, *Focus on K. for LSTM1 Output Gate of $m=0 \& s=2$* and *Focus on K. for LSTM1 Output Gate of $m=0 \& s=3$* for the output gate, as well as *Focus on K. for LSTM1 Input Gate of $m=0 \& s=2$* and *Focus on K. for LSTM1 Input Gate of $m=0 \& s=2$* for the input gate. Activations of all the other moments are considered as $-kb$ and refer to the disregarding of activations. With this, all the cases the input neuron must deal with can be explained plausibly. Right then the kind of knowledge pattern is produced that is required at consecutive neurons. Regarding the input objects of that activation, the explicit knowledge forms are considered that are injected in the ANN. Hence, a neuronal internalization can be identified here. Beside the *Arousal* values presented at the corresponding model of the CoNM repository, its values can be found at Fig. 6.22 first column of the input section).
8. *Activation of Input 2 Neuron:* For the activation of the network, further, the input neuron called *Inp2* provides the focus knowledge $+kb$ for the three kinds of LSTM gates and its gated LSTM cell at the first moment of the second and fourth scenario only. This refers to the relevant XOR activation, and corresponding knowledge objects are labeled by *Focus on K. for LSTM1 Forget Gate of $m=0 \& s=1$* and *Focus on K. for LSTM1 Forget Gate of $m=0 \& s=3$* for the forget gate, *Focus on K. for LSTM1 Cell1 of $m=0 \& s=1$* and *Focus on K. for LSTM1 Cell1 of $m=0 \& s=3$* for the LSTM cell, *Focus on K. for LSTM1 Output Gate of $m=0 \& s=1$* and *Focus on K. for LSTM1 Output Gate of $m=0 \& s=3$* for the output gate, as well as *Focus on K. for LSTM1 Input Gate of $m=0 \& s=1$* and *Focus on K. for LSTM1 Input Gate of $m=0 \& s=2$* for the input gate. Activations of all the other moments are considered to be $-kb$ and refer to the disregarding of activations. With this, all the cases the input neuron must deal with can be explained plausibly. Right then the kind of knowledge pattern is produced that is required at consecutive neurons.
- Regarding the input objects of that activation, the explicit knowledge forms are considered that are injected in the ANN. Hence, a neuronal internalization can be identified here. Beside the *Arousal* values presented at the corresponding model of the CoNM repository, its values can be found at Fig. 6.22 second column of the input section).

Regarding the knowledge objects identified, one can record that the behavior interpreted corresponds to the design intention of LSTM blocks (Hochreiter and Schmidhuber, 1997b).

Similar to the visualization of structural knowledge objects by the NMDL *KnowledgeGenerationView*, (see Fig. 6.24 and Fig. 6.25), dynamic knowledge objects can be visualized. Using this, the knowledge flow as well as neuronal conversions become clear based on real knowledge values such that the transparency of the ANN can be increased and the generation, transfer and application of knowledge becomes transparent. As this does not generate more insights than the ones presented so far, and the fact that the NMDL *KnowledgeGenerationView* has been demonstrated once, these kinds of models have not been addressed any further in the following.

Communication: On behalf of the DKI, the interpretation of the ANN structure was enabled. Knowledge objects identified made the concrete signaling transparent so that it became clear as to which kind of signals are relevant from the viewpoint of a neuron. This includes both, the relevance for the knowledge receiving and the relevance for the knowledge transmission. Particularly in terms of the characteristic to function exciting or inhibiting (edge color) and the influence of signals because of the weights (edge thickness), the interpretation of knowledge objects identified have been relativized plausibly.

This proves that the NPM enables the plausible interpretation of ANN and rises the transparency in terms of crystallized knowledge codified within the ANN. Further, this gives evidence of having identified tacit knowledge with the aid of the NMDL.

Contribution: In context of the CoNM, the DKI functions as a tool whitening the black box of ANN with signal-based knowledge. It further enables cooperative modeling and symbiotic KM (section 4.5) of human and ANN-based model constructors.

6.5 Neuronal Process Simulation

The following experiments intend to demonstrate the functioning of the CoNM with respect to the third sub-research question, that is, (*How can the modeling of neuronal processes be used for process simulations?*), which can be found in section 1.2. As the NPS considers interconnected ANN systems as joint ANN, the following focuses on the demonstration of simulation capabilities of the CoNM designed in section 4.8.2.

Although this research endeavors to demonstrate important views once, the detailed views, their technical linking via shortcuts and the navigation throughout all models can be seen in the corresponding CoNM project and its public repository.

6.5.1 Experiment 4—Time-Oriented Process Simulation

This experiment focuses on the demonstration of the simulation capabilities of the CoNM to consider ANN to be knowledge carriers within their processual context of business processes. It considers simulations to be a time-oriented method (cf. section 2.3.2.10), which means the following:

Simulation time and ANN activation are in sync.

So, one simulation step corresponds to the activation of corresponding ANN structures. Coming to live simulations, the experiment shows how to deal with the data provision of live systems.

Further, the experiment demonstrates the NMDL visualization capabilities to function as a communication platform for the specification of simulation runs and the presentation of simulation result. It enables cooperative modeling and process-oriented KM by both, human and machine-based model creators as it was designed by symbiotic KM approach in section 4.5. For instance, a simulation scenario and ANN design is set up by a human model creator. This is considered by the neuronal system carrying out NPS such that the models constructed are complemented by the ANN simulation system.

Problem Identification and Motivation: As ANN are considered tools for the arbitrary detailed system approximation that establish a behavior which unfolds within its processual context, one can identify the potential of trained ANN to function as simulation elements in process simulations.

So far, ANNs have been considered to be decoupled and isolated systems approximating a trained behavior (cf. section 2.7). Provided one wants to talk from a simulation, this refers to the least demanding setting for a neuronal simulation. Here, one must understand that ANN structures have been considered by a simulation system (compare with Fig. 2.31). Although even this shows the great approximation capabilities of ANN, even for their dynamic effects because of recurrent connections, this does not address their integration as knowledge carriers within business processes. Further, because of a missing integrative definition, neither the potential of ANN to function in simulators as individual systems nor the potential of ANN to function as simulation systems have been addressed appropriately.

Systematized by the taxonomy developed in Fig. 2.36, the following evaluates limitations of the contemporary ANN use in the simulation context by the taxonomic classes or rather dimensions which have been italicized. ANN systems have neither been considered in live process simulations (*real time dimension*), considering hybrid symbiotic relations with real systems (*symbiosis dimension*) allowing to integrate virtual and real components (*digitalization dimension*), nor have they been considered deterministic simulation systems (*behavior dimension*) operating with discrete values (*basis of value dimension*), manifesting as dynamic simulation system (*presence of time dimension*) and being a feedbacking simulator because of recurrent connections (*feedbacking dimension*). Hence, ANN systems have further a potential to function, for instance, as agents (*specialization dimension*) in non-terminating (*termination dimension*) and non-stationary simulations (*stationary dimension*) that are time-oriented (*time advancement dimension*).

However, since neither a neuronal simulation system was set up nor this was modeled by a modeling language, a research gap can be identified in the description of how ANN are integrated with business processes and simulation systems.

Objective of the Solution: Based on the understanding of ANN to function as *neuronal process simulation* systems (cf. section 4.8.2), a computational approach was designed to enable ANN to be coupled flexibly on an arbitrary detailed level of system specification. This allows ANN to be coupled flexibly so that an independent overlapping of their function as actual entities is enabled and an arbitrary detailed level of systems can be specified. The approach designed enables the following two: first, ANN systems being part of a simulation; and second, the simulation system being an ANN holding ANN systems.

With the aid of the modeling language *NMDL* designed, any such system can be specified. Further, with the NMDL, the scenarios can be specified as to with whom the simulations will be carried out.

A simple experiment setting was designed for the demonstration of NPS, because even the smallest ANN provides thousands of elements and parameters. By this, a clear model construction as well as its straight-forward interpretation is supported, and the functioning of the NPS becomes clear.

Experimental Design: By following the CoNM proceeding (section 4.4), a BP was set up, which is shown in Fig. 6.27. This refers to a process that is part of the *Research and Application Center for Industry 4.0* (RACI4.0), in which either a production machine called *Production Machine 1* demands a transportation in three time steps, or a production worker called *P1* announces a conveyor maintenance in three time steps. The conveyor object called *CPS1* perceives both kinds of information objects.

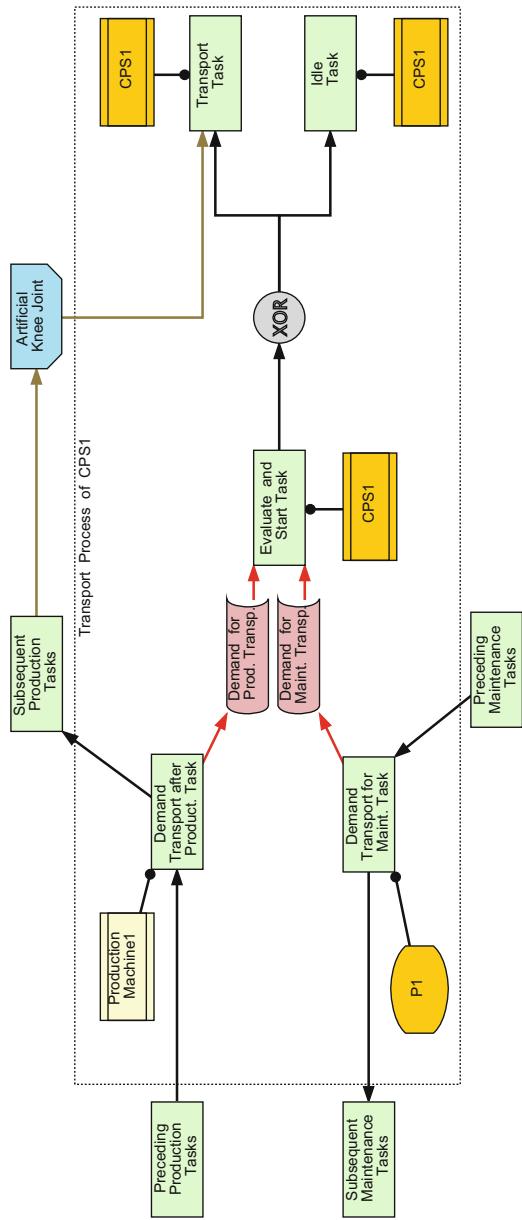


Figure 6.27 The process to be simulated for the NPS Demonstration (*ProcessView*)

It evaluates for the machine if the task called *Transport Task* shall be initiated so that either the tangible object of an *Artificial Knee Joint* can be transported or the production worker can check the transport capabilities of the CPS. Alternatively, the *Idle Task* is realized, which causes the stopping of the CPS's actuators. These are either stopped because no demand is available or because the CPS must care about conflicting demands. Please note the system border which only holds the process steps that need to be considered by the simulator.

Following the meta-model designed (cf. Fig. 4.4), one can identify the process network use of the simulation scenarios designed in Fig. 6.28(a). By looking at the positioning of the scenario called *Production but No Maintenance Demand* in the system border at the corresponding *SimulationOverview*, the NPS is instructed to simulate the selected scenario. Because of the hierarchical decomposition of process networks, processes, tasks, activities and activations visualized by the corresponding overviews in Fig 6.28, it becomes clear as to which modeling elements need to be simulated and which neurons must be activated by the NPS. One can identify the hierarchical decomposition of process networks to processes from Fig. 6.28(b), of processes to tasks in Fig. 6.28(c), as well as the consideration of knowledge bearers at tasks by the breakdown of tasks to activities shown in Fig. 6.28(d), activities to activations in Fig. 6.28(e) and their scaling down to neurons in Fig. 6.28(f).

Since the activity called *Evaluate Request and Start Transportation Activity* is based on the activation called *Activation of LSTM Neurons 1*, as visualized in Fig. 6.28(e), one can identify the reuse of the LSTM model that has already been trained during the previous experiment (section 6.3.1). This has been imported from the CoNM Model Warehouse, and in this example, it unfolds the XOR behavior of the *CPS1* trained during the task called *Evaluate and Start Task* (see Fig. 6.27). This had been recognized by the analysis of knowledge objects identified by SEKO (see experiment of section 6.5).

As Fig. 6.27 further shows, *CPS1* interacts with the ANN systems of *P1* and *Production Machine 1* by the preceding tasks. Coming to its subsequent tasks, one can see that *CPS1* interacts with its actuator ANN. In general, to what extent ANN systems will interact has been specified by the joint set of flow-oriented NMDL views in Fig. 6.29. For instance, the knowledge-based interaction of their activities has been specified by the *ActivityView* in Fig. 6.29(d). Their interactions on the level of neuronal activations can be found in Fig. 6.29(e). To what extent ANN systems interact within a network of processes has been specified by the *Network View* in Fig 6.29(b). Please note here the system border which specifies that some network processes are not part of the NPS. Further, as the selected network setting of Fig. 6.29(a) shows, the NPS focused only on the simulation of one scenario.

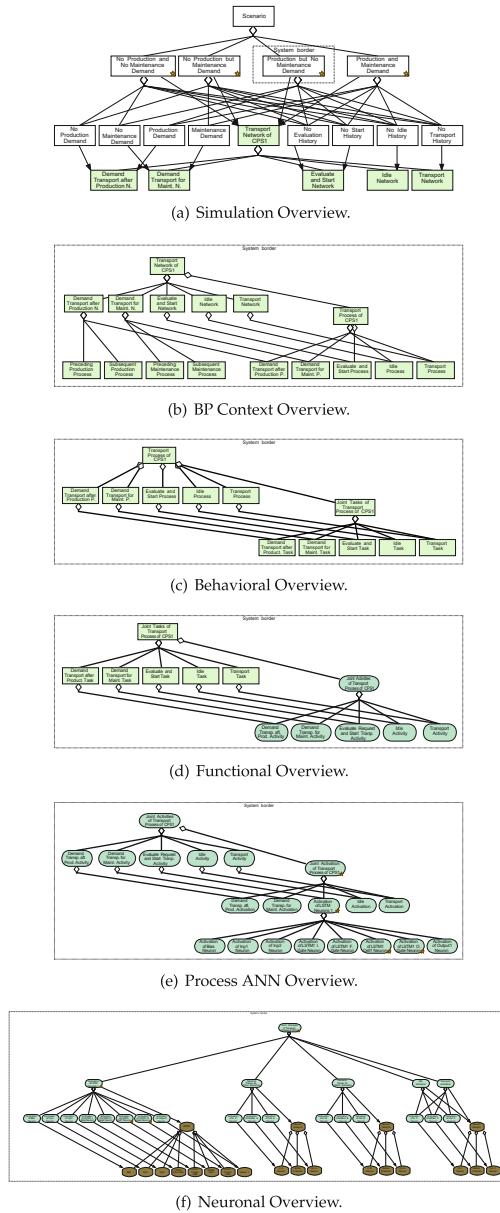
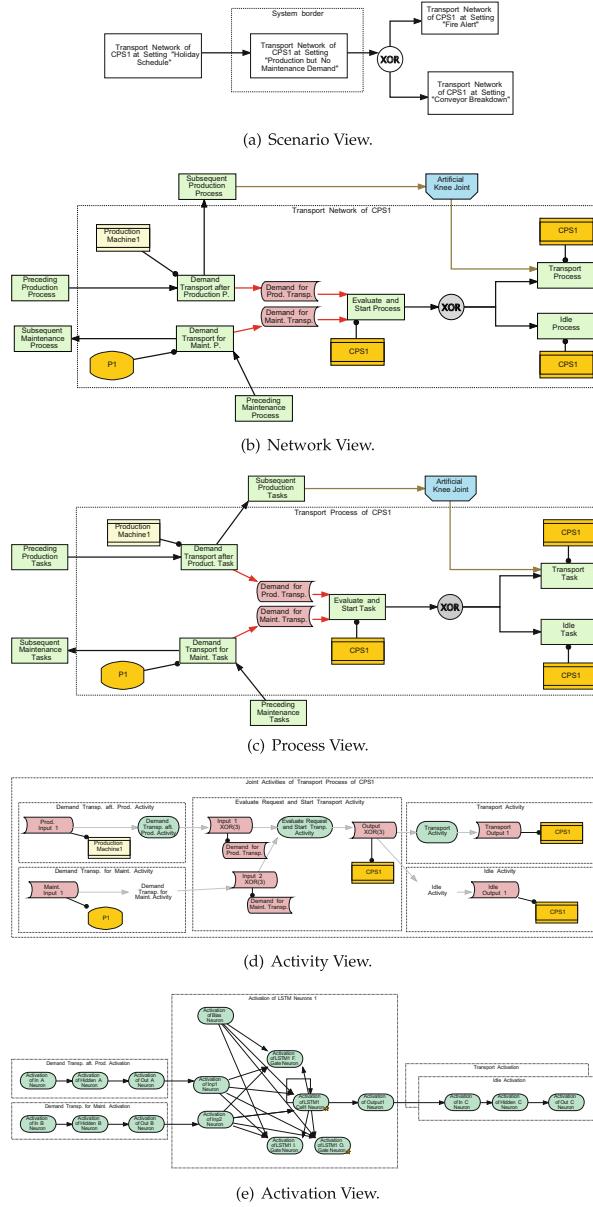


Figure 6.28 The Overviews of NPS Example

**Figure 6.29** The Views of NPS Example

Training Phase: Considering the joint overviews of Fig. 6.28 and the joint flow-oriented views of Fig. 6.29, the simulation system was set up on a neuronal level such that Fig. 6.30 clarifies the neuronal structures.

In the figure, one can see that four kinds of ANN have been put side by side. This corresponds to the specification of the *ActivationView* in Fig. 6.29(e). This becomes clear as each individual ANN has been surrounded by an activation-specific system border having the same label as its individual ANN. Since these have been connected by pragmatic knowledge-flow edges (black colored edges in Fig. 6.30), the joint neuronal simulation system has been set up on a neuronal level.

Since the LSTM network already has been trained by a previous experiment, the ANN only had to be imported from the *Model Repository* of the CoNM (cf. reference models, Def. 20). In the sense of a cooperative modeling, its surrounding systems have been constructed manually so that individual training procedures are not required.

Testing Phase: Regarding the scenario's attribute called *SetView Model* of the *SimulationView* in Fig. 6.28(a), the scenario-based activation for the joint neuronal simulation system has been specified. As the atomic scenarios of the figure show, each process network starts at a clean ANN. Otherwise, before the neuronal simulation starts, the historic activation of each ANN component would have had to be prepared.

The final dataset constructed manually has been labeled as *Production but No Maintenance Demand* because it seemed plausible for the model constructor. The corresponding dataset can be found in Fig. 6.31(a). Although this dataset seems similar to the dataset specified in Fig. 6.18, the scenario-based set is individual as the CoNM mechanisms write back simulation results at the output variables. So, by storing the models at the *Simulation Warehouse* (cf. Fig. 4.8), simulation results are stored and can be reused in third-party simulators. Further, they can be used for NPM such as SEKO concepts demonstrated in section 6.4.

Since the neuronal simulator has been set up as a live system, the example demonstrates the inclusion of the production worker via an input workflow; its input has been read live during the simulation from the location specified at the knowledge object's attribute called *Workflow Input* in Fig. 6.31(a). This input has been documented automatically at the corresponding attribute called *Knowledge Value*.

Moreover, as the neuronal simulator has been set up as a hybrid simulation system, the CPS of the RACI4.0 has been integrated as third-party systems via an output workflow; the NPS output has been written live during the simulation at

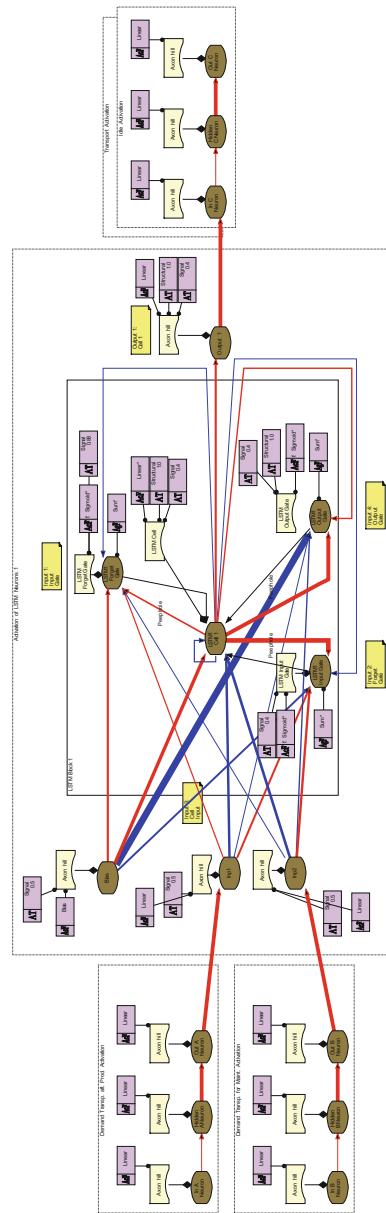


Figure 6.30 The Training Specification for the NPS Demonstration (*NeuronView*)

the location specified at the knowledge object's attribute called *Workflow Output* in Fig. 6.31(a).

Evaluation: The simulation results of the NPS have been visualized by Fig. 6.31(b). Since the ANN structures surrounding the LSTM network were designed to route input only, the NPS output corresponds to the LSTM output. The NPS performance can be derived by the deviation of the NPS output results from the correct behavior expected. Hence, the deviation can be indicated by the kind of error measurement selected.

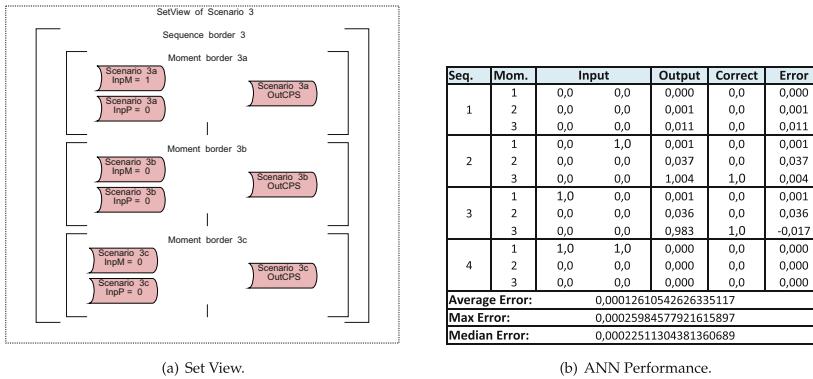


Figure 6.31 The Performance of the ANN Trained

Communication: On behalf of the NPS, the interpretation of the ANN within its processual context was enabled. The ANN simulated manifested concrete behavior and knowledge so that it became clear as to which kind of systems are relevant from an organization's point of view. Particularly, in terms of the characteristic feature to function as a representation of the organization's current processes and resources to be managed, the potential as a management tool for a process-oriented, symbiotic KM has become transparent.

Further, this gives evidence that the NPS enables the plausible interpretation of ANN-based BP and enables the neuronal optimization of BP.

Contribution: In context of the CoNM, the NPS functions as a tool to consider ANN as knowledge carriers and to jointly activate the entire organizational knowledge transfer model. This shows that it has the potential to function as neuronal enterprise architecture. It further enables cooperative modeling and symbiotic KM (section 4.5) between human and ANN-based model constructors, as it enables the use of an organization-wide knowledge base within the processual context of scenarios specified.

6.6 Neuronal Process Optimization

The following experiments intend to demonstrate the functioning of the CoNM with respect to the fourth sub-research question, that is, (*How can the learning capability of NN systems be used for process optimizations?*), which can be found in section 1.2. As the NPO considers the simulation system to be joint ANN, the following focuses on the demonstration of learning principles of the CoNM designed in section 4.8.

Although this research endeavors to demonstrate important views, the detailed views, their technical linking via shortcuts and the navigation throughout all models can be seen by the corresponding CoNM project and its public repository.

6.6.1 Experiment 5—NPO Backpropagation Procedures

This experimentation focuses on the demonstration of the *NPO Backpropagation* training that addresses the target-oriented weight modification of higher cognitive super systems and has been designed in section 4.8.3.

Further, the experiment demonstrates the NMDL visualization capabilities to function as a communication platform for the specification of optimization runs and the presentation of optimization result. It enables the cooperative modeling and process-oriented KM by both, human and machine-based model creators, as shown by the symbiotic KM approach in section 4.5. For instance, an optimization run and ANN design is set up by a human model creator. This is considered by the neuronal system carrying out NPO so that the models constructed are complemented by the ANN optimization system.

Problem Identification and Motivation: As ANN are considered to be tools for the arbitrary detailed system approximation that establish a behavior which unfolds within its processual context, one can identify the potential of biologically plausible learning approaches of ANN to optimize processes.

So far, ANN have been trained by error-based (Plaut, Nowlan, and Hinton, 1986; Bishop, 1995), competitive (Barlow, 1989; Ritter, Martinetz, and Schulten, 1991, p. 68; as well as Rolls and Deco, 2001) and Hebbian (Hebb, 1949; Oja, 1982; as well as Sanger, 1989) learning algorithms (cf. section 2.5.6). Here, training algorithms are used as tools to modify ANN so that these satisfy performance requirements for individual tasks. Although even this shows the great learning capabilities of ANN - even for temporal patterns because of learning through the time - these have not been considered to be systems representing process networks. So, learning approaches have not overcome the activation border of one isolated training task identified here. Because of a missing integrative definition of ANN learning and BP, neither the respecting of individual ANN to be knowledge carriers, nor the learning of greater cognitive structures, such as process networks, processes, tasks, activities and activations, have been addressed appropriately.

Further, a neuronal optimization was not modeled by a modeling language, such that a further research gap can be identified in the description of how ANN are integrated with business processes and optimization systems.

Objective of the Solution: Based on the understanding of ANN to function as process approximation (for instance of continuously improving learning organizations) established in section 4.8.3, a computational approach was designed to enable ANN to be optimized flexibly on an arbitrary level of system specification. This allowed ANN to be trained flexibly so that an independent overlapping of their function as actual entities could be enabled and an arbitrary detailed level of system optimization could be specified. The approach designed enabled the following two things: first, respecting ANN systems being part of a simulation; and second, the optimization of super ANN holding ANN systems.

With the aid of the modeling language *NMDL* designed, any such system optimization can be specified. Further, using the NMDL, *NPO Backpropagation* can be set up to carry out optimization.

A simple experiment setting was designed for the demonstration of NPO, because even the smallest ANN provides thousands of elements and parameters. From this, it can be seen that a clear model construction as well as its straight-forward interpretation is supported and the functioning of the NPO becomes clear.

Experimental Design: By following the CoNM proceeding (section 4.4), a BP was set up, as presented in section 6.5.1. This refers to a process part of the RACI4.0, in which either a production machine called *Production Machine1* demands a transportation in three time steps, or a production worker called *P1*. The CPS called *CPS1* evaluates both demands and instructs its conveyor actuators in time.

Training Phase: The training task has been specified in two ways. First, an ANN trained at the XOR dataset will be faced by the *NPO BackpropTrainer* with the dataset called *XOR*. The ANN and dataset had been constructed at the Live Learning experiments (see Fig. 6.17 and is shown in Fig. 6.13) and thus refers to a fixed learning task according to Def. 21. The learning task described here can be seen in Fig. 6.32(a). So, it can be interpreted as the standard environment the ANN was trained for. Hence, during the course of NPO learning, the ANN is intended to carry out only negligible small changes so that the ANN can operate without behavioral changes, without its knowledge base getting affected.

Second, the very same ANN will be faced by the *NPO BackpropTrainer* with a training and a test dataset labeled as *Dataset NPO*. These datasets have been set up with 10 sequences, each holding one moment that corresponds to the dataset called *XOR*. It is different from the XOR dataset mentioned in that the second input refers to a random number of either 0.0 or 1.0 not affecting the output at all. So, it can be interpreted as noise or defect input channel disturbing the ANN. Hence, during the course of NPO learning, the ANN is intended to shrink the influence of the disturbing process step of that input channel. If the ANN recognizes to disregard this input channel and to adapt to the environmental change reflected in the noisy data provided, the current BP performance is optimized. The corresponding learning task can be seen in Fig 6.32(b).

For ANN specification, the models were imported, which have been presented in the NPS experiment shown in section 6.5.1. Since the *Dataset XOR* has been selected for NPO demonstration, the activation called *Activation of Neuron Group 1* was associated with the activity called *Evaluate Request and Start Transp. Activity* shown in Fig. 6.28(e). The ANN so replaced the LSTM block with the ANN trained in section 6.3.1, as shown in Fig. 6.17. Thus, the ANN that was used for the training with new learning principles can be found at Fig. 6.33.

Testing Phase: The NPO was run for 100 training sessions for one training run per session. This refers to the stopping strategy called *Train Once* (see Fig. 6.32). The training and testing performance for the learning tasks can be found in Fig. 6.34.

Here, one can identify that the best training and test performances refer to the last training sessions without violating the test and training error performance requirements specified. The corresponding ANN structures of the *NetworkView* can be found in Fig. 6.35.

The NPO modifications of the first learning task can be seen in Fig. 6.35(a). Please note the stroke width of the three edges that cross the system borders of ANN systems. They can be interpreted as follows: The interconnecting neuronal weights have not been modified much. They refer to the value of 1.0 having only a

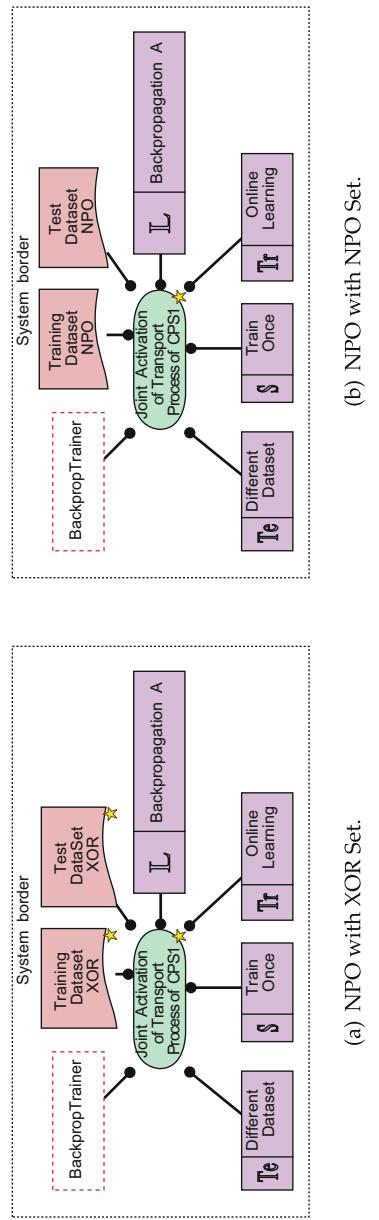


Figure 6.32 The Training and Testing Specification for the NPO Demonstration (*Activation View*)

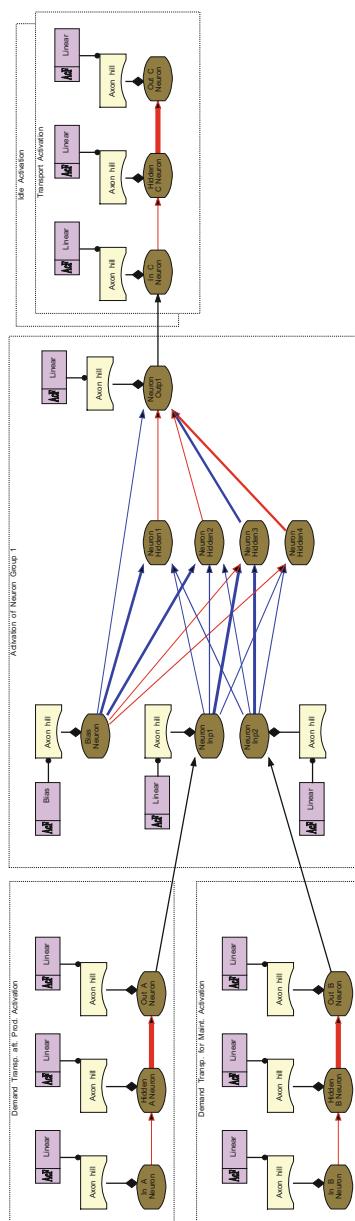
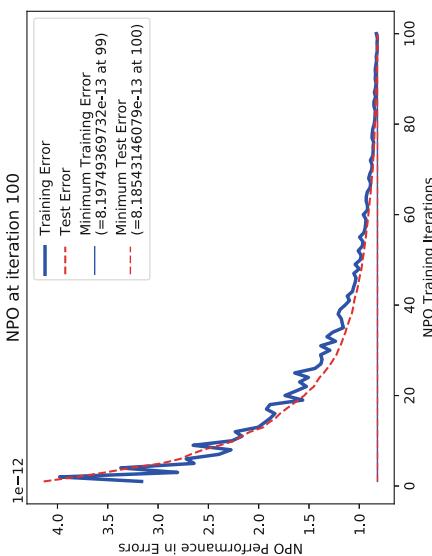
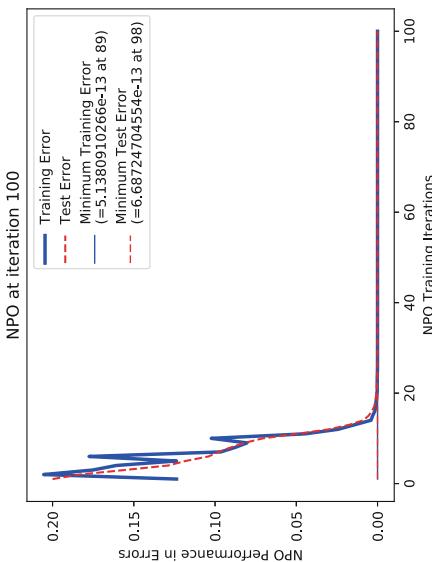


Figure 6.33 The Initial Network Structure for the NPO Demonstration of NPO Backpropagation Procedures (Experiment 5) and Standard Backpropagation Procedures (Experiment 6)



(a) NPO with XOR Set.



(b) NPO with NPO Set.

Figure 6.34 The Course of Training and Testing for the NPO Demonstration of *NPO Backpropagation* Procedures

thousandth of deviation. Hence, the NPS will lead to the same performance, which already has been accepted before. The NPO modifications of the second learning task can be seen in Fig. 6.35(b). Please note here the stroke width of the same three edges that have been focused before. They can be interpreted as follows:

1. The first input is considered further, which can be seen at the thick weight between the neuron called *Out A Neuron* and the neuron called *Neuron Inp1*. Here, the latter intends to consider the input of 1.0 around the level of 1.0 and the input of 0.0 around the level of 0.0.
2. The influence of the second input is diminished by the ANN. Between the neuron called *Out B Neuron* and *Neuron Inp2*, one can only identify a slight line. Here, the latter neuron intends to consider the input of 1.0 around the level of 0.0 and the input of 0.0 around the level of 0.0 as well. This is the desired behavior because the second input referred to unusable noise data coming from an inefficient preceding task.
3. The influence of the output is considered further, as a thick edge can be identified between the neuron called *Neuron Outp1* and the neuron called *In C Neuron*. Here, the latter intends to consider the input of 1.0 around the level of 1.0 and the input of 0.0 around the level of 0.0.

Since the NPO only affects edges interconnecting more complex systems, which is in this case the activation called *Joint Activation of Transport Process of CPSI*, further weights have not been modified. Hence, by disregarding the influence of the noisy second input channel, the NPS will perform well at the same performance level which has already been accepted before.

Evaluation: Based on the weight modifications described, one can derive the process change of pruning the irrelevant input process either by human model creators or algorithmically by machine-based model creators. As the network has not changed any further, the very same performance level can be achieved as was achieved by the NPS described in section 6.5.1.

Communication: On behalf of the NPO, the optimization of the ANN within its processual context was enabled. The ANN trained by the *NPO BackpropTrainer* have lead to a different manifestation of the concrete behavior and knowledge so that it becomes clear as to which kinds of systems are relevant from the viewpoint of an organization. Particularly, in terms of the characteristic to function as representation of the organization's current processes and resources to be managed, the potential

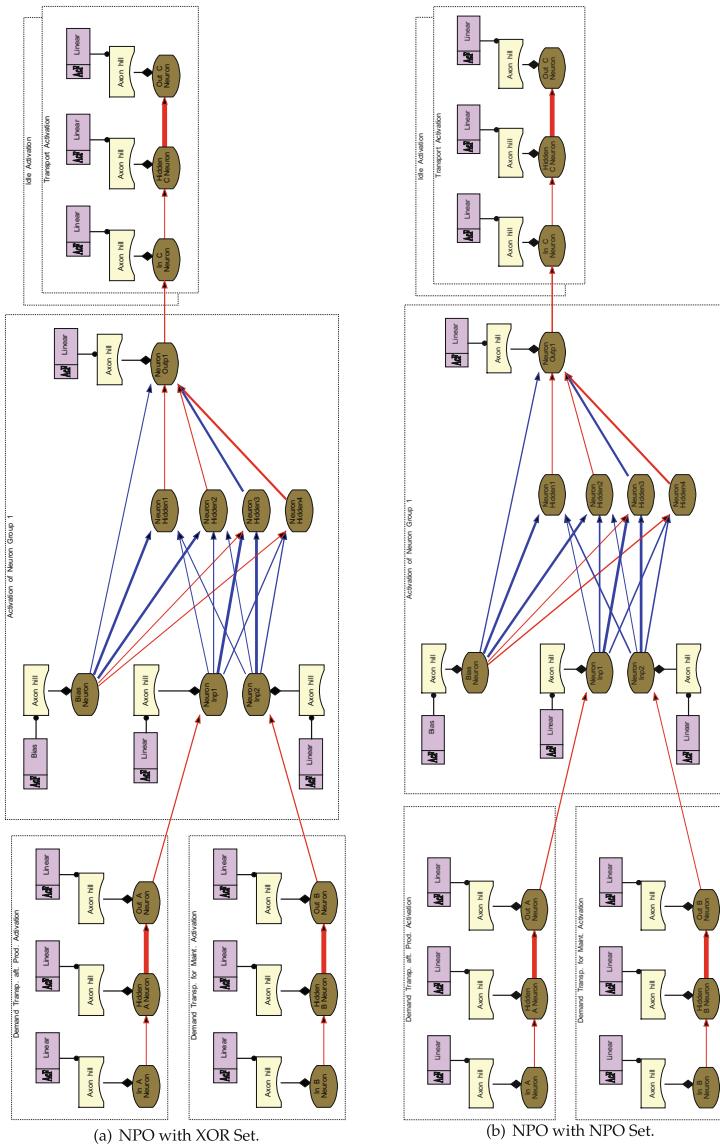


Figure 6.35 The Structural Results for the NPO by *NPO Backpropagation* Procedures Demonstration (*NeuronView*)

as a tool for a continuously improving, process-oriented, symbiotic KM has become transparent.

Further, this gives evidence that the NPO enables plausible improvement of ANN-based BP and neuronal optimization of BP.

Contribution: In context of the CoNM, the *NPO Backpropagation* functions as a tool to consider ANN as knowledge carriers and to jointly improve the entire organizational knowledge transfer model. This shows that it supports the potential to function as a learning neuronal enterprise architecture.

It further enables the training of greater cognitive levels, such that a multidimensional ANN architecture can be established by employing pedagogic designs of the symbiotic KM (section 4.5). Hence, it enables the cooperative modeling and management by human and ANN-based model constructors, as it operationalizes the use of an organization-wide knowledge base within the processual context of scenarios specified.

6.6.2 Experiment 6—Standard Backpropagation Procedures

This experimentation focuses on the demonstration of standard *Backpropagation* procedures for NPO training. This addresses the weight modification of any kind of ANN system and has been presented in section 4.8.3.

Since this experiment can be interpreted as a variant of the experiment described in section 6.6.1, the following addresses only relevant changes.

Problem Identification and Motivation: Having successfully modified the ANN interconnections, one might desire to further improve the joint ANN performance in NPS runs. This demands for the modification of any neuronal weight of the joint ANN.

Objective of the Solution: Based on the understanding of ANN to function as process approximation, such as for continuously improving learning organizations, as established in section 4.8.3, a computational approach was designed that enables ANN to be optimized flexibly on all levels of system specification. This allows ANN to be trained flexibly, such that an independent overlapping of their function as actual entities is enabled and an arbitrary detailed level of system optimizations can be specified. The approach designed enables the following two: first, respecting ANN systems being part of a simulation; and second, the optimization of super ANN holding ANN systems.

With the aid of the modeling language *NMDL* designed, any such system optimizations can be specified. Further, using the NMDL, the standard *Backpropagation* procedures for NPO can be set up, using which the optimization can be carried out.

Training Phase: The training task for generating NPO modifications on any neuronal level has been specified in two ways. First, an ANN will be faced by the *BackpropTrainer* with the dataset called *XOR*; so, it can be interpreted as the standard environment the ANN was trained for. Hence, during the course of learning, the ANN is intended to change just a bit so that it can operate without behavioral changes.

Second, the very same ANN will be faced by the *BackpropTrainer* with a training and a test dataset labeled as *Dataset NPO*. Hence, the very same learning tasks, datasets and ANN are addressed that have been presented in section 6.6.1.

Testing Phase: Again, the NPO was run for 100 training sessions for one training run per session. This refers to the stopping strategy called *Train Once* (see Fig. 6.32).

The training and testing runs lead to ANN structures, which have been visualized by the NMDL *NetworkView*. These can be found in Fig. 6.36.

The NPO modifications of the first learning task can be seen in Fig. 6.36(a). Please note the stroke width of the three edges that cross the system borders of ANN systems. They can be interpreted as follows: The interconnecting weights have not been modified much. They refer to the value of 1.0 having only a thousands of deviation. Hence, the NPS will lead to at least the same performance level which already has been accepted before the NPO training. Compared with the ANN in Fig. 6.35(b), one can recognize the changes in inner ANN weights as well. This for instance is attractive, if one aims to make the ANN evolve in its greater processual contexts.

The NPO modifications of the second learning task can be seen in Fig. 6.36(b). Please note here the stroke width of the same three edges that have been focused on before. They can be interpreted as follows:

1. The first input is considered further, which can be seen at the thick weight between the neuron called *Out A Neuron* and the neuron called *Neuron Inp1*. Here, the latter intends to consider the input of 1.0 around the level of 1.0 and the input of 0.0 around the level of 0.0.
2. The influence of the second input is diminished by the ANN. Between the neuron called *Out B Neuron* and *Neuron Inp2*, one can only identify a slight line. Here, the latter neuron intends to consider the input of 1.0 around the level of 0.0 and the input of 0.0 around the level of 0.0 as well. This is the desired behavior because

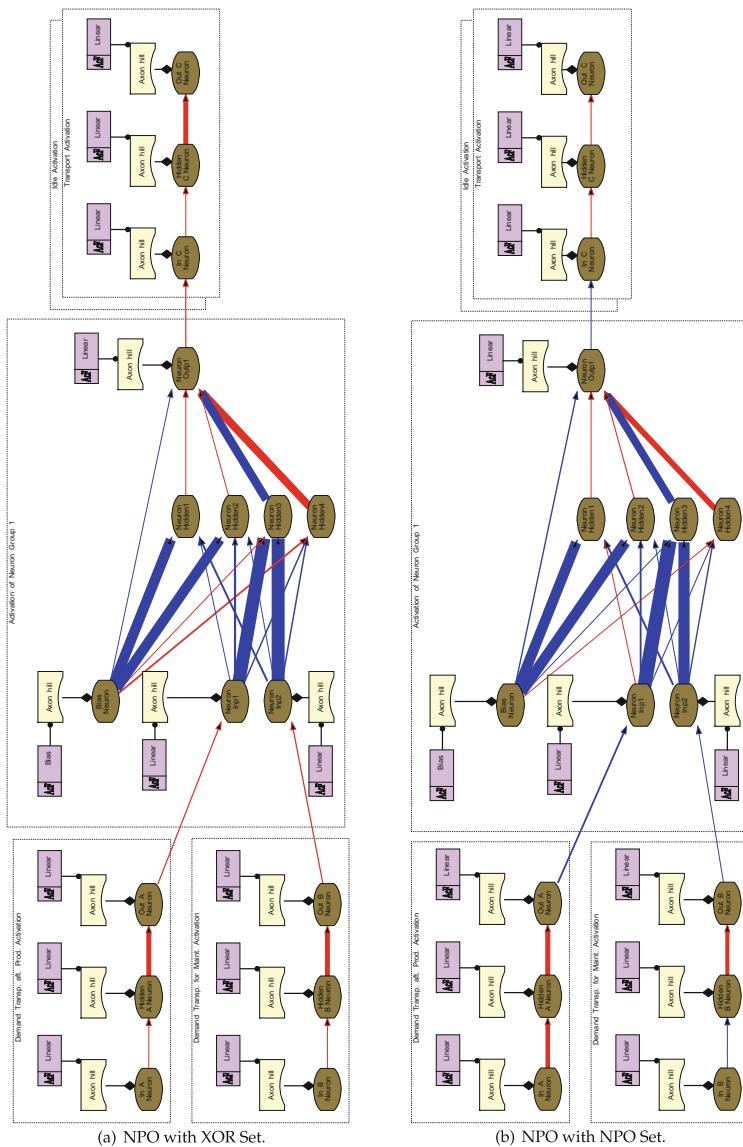


Figure 6.36 The Structural Results for the NPO by Standard *Backpropagation* Procedures Demonstration (*NeuronView*)

- the second input referred to unusable noise data coming from an inefficient preceding task.
3. The influence of the output is considered further, as a thick edge can be identified between the neuron called *Neuron Outp1* and the neuron called *In C Neuron*. Here, the latter neuron intends to consider the input of 1.0 around the level of 1.0 and its input of 0.0 around the level of 0.0.

Since the NPO does not only affect the edges interconnecting more complex systems, which is in this case the activation called *Joint Activation of Transport Process of CPSI*, the joint ANN has been modified. Hence, by disregarding the influence of the noisy second input channel, the NPS will perform at least at the same performance level which already has been accepted before. Compared with the ANN in Fig. 6.35(b), the structural changes in inner ANN weights can be identified. This for instance is attractive if one aims to make the ANN evolve in its greater processual contexts.

Evaluation: Based on these modifications, one can derive the process change of pruning the irrelevant input process either by human model creators or algorithmically by machine-based model creators. As the network has not changed any further, the very same performance level can be achieved as was described by the NPS in section 6.5.1.

Communication: On behalf of the NPO, the optimization of the ANN within its processual context was enabled. The ANN trained by the NPO with standard *Backpropagation* procedures have lead to a different manifestation of concrete behavior and knowledge, such that it becomes clear as to which kinds of systems are relevant from the viewpoint of an organization. In comparison to the NPO with *NPO Backpropagation* procedures, inner weights have been modified as well. Particularly, in terms of the characteristic to function as representation of the organization's current processes and resources to be managed, the potential as a tool for a continuously improving, process-oriented, symbiotic KM has become transparent, which considers any neuronal compartment.

Further, this shows that the NPO enables plausible improvement of ANN-based BP and neuronal optimization of BP.

Contribution: In context of the CoNM, the NPO by standard *Backpropagation* procedures functions as a tool to consider ANN to be knowledge carriers and to jointly improve the entire organizational knowledge transfer model. This shows that it supports the potential to function as learning neuronal enterprise architecture.

It further enables the training of greater cognitive levels, such that a multidimensional ANN architecture can be established by employing pedagogic designs of the symbiotic KM (section 4.5). It further enables the cooperative modeling and management by human and ANN-based model constructors, as it operationalizes the use of an organization-wide knowledge base within the processual context of the scenarios specified.

6.7 Conclusion about Demonstrations

The demonstrations presented intend to stand as a proof-of-concept for the functioning of the CoNM. Based on a short summary for each demonstration, key findings have been collected and interim conclusions have been presented. Finally, a conclusion about the demonstrations follows.

Interim Conclusions: The first demonstration (section 6.1) has characterized the generic evaluation and selection system (short: GESS, cf. section 4.2). Based on its special version of CoNM-specific evaluation and selection system (CoNM-SESS), an appropriate technical foundation of the CoNM construction was identified. The modeling tool called *Modelangelo* and the programming library called *PyBrain* served as the foundation for the CoNM implementation. Among others, *ARIS* served as a conceptual design input for an architectural inspiration. The *Potsdam KM Model* inspired the symbiotic KM model design, and the *Space Markup Language* was inspired by a geometrical modeling space. Summarizing, this demonstration has achieved the following:

- ⇒ The CoNM-specific evaluation and selection system was set up.
- ⇒ An appropriate technical foundation for the CoNM construction was identified in terms of an aspect-wise performance and popularity.

The second demonstration (section 6.2) has presented prominent ANN architectures with the help of the NMDL. These referred to individual neurons, Multilayer Perceptrons, feedforward and recurrent networks as well as Long-Term-Short-Term-Memory blocks. These can serve as building blocks for neuronal modeling in the sense of the CoNM and as reference models in accordance with Def. 20, using which the experiments were realized. Further, neuronal conversions were modeled, which formed the base for neuronal process modeling. Summarizing, this demonstration has achieved the following:

- ⇒ Prominent ANN architectures have been transcribed into the NMDL.
- ⇒ Prominent ANN architectures have been set up as reference models.
- ⇒ Conversions were transferred to the neuronal level.

The third demonstration (section 6.3) shows the design of ANN-based systems with the help of the CoNM artifacts. Due to the functioning of the NMDL as a communication platform for human as well as machine-based model constructors, the NMDL has demonstrated the capability of constructing models cooperatively. First, the ANN design has been graphically set up by the human model constructor. Then, the ANN-based systems have visualized inner workings in the very same graphical model. So, both kinds of model creators have constructed models in an iterative cooperation. In the first experiment, focus has been set on the specification of an ANN training and the visualization of structural changes of the ANN compartments. So, the black-box of ANN is whitened. Summarizing, this demonstration has achieved the following:

- ⇒ The ANN learning task has been set up by the NMDL.
- ⇒ Training and testing of ANN have been realized by CoNM tools. Here, the modeling was realized with the aid of the extended modeling tool called *Modelangelo*. The training was realized on the base of the programming library called *PyBrain* and individual CoNM implementations.
- ⇒ The learning procedure has been visualized by the NMDL in terms of structural ANN changes in run-time. The transparency has increased because neuronal weights have been visualized through association thickness, and the kind of arousal has been visualized through the color of corresponding associations.

The fourth demonstration (section 6.4) has depicted the transparency increase of ANN-based systems by focusing on the identification of tacit knowledge. While the second experiment has issued the extraction of static knowledge, the third experiment has issued the search for a pattern-based, dynamic knowledge. The behavior of the ANN-based system has been explained successfully with the aid of knowledge objects extracted by the CoNM algorithms because the objects of both experiments have lead to plausible interpretations of the joint ANN-based system. The experiments have clarified the capability of the NMDL to be the base for a symbiotic knowledge management because of the successful integration of traditional and neuronal knowledge management approaches with the CoNM. Knowledge objects extracted from human as well as machine-based knowledge carriers can be accessed

and managed right within their processual context. Summarizing, this demonstration has achieved the following:

- ⇒ Static knowledge has been identified in ANNs by CoNM mechanisms.
- ⇒ Dynamic, pattern-based knowledge has been identified in ANNs by CoNM mechanisms.
- ⇒ Explicit and tacit knowledge have been distinguished on a neuronal level.
- ⇒ A neuronal knowledge base has been set up which complements the organizational knowledge base.

The fifth demonstration (section 6.5) shows the realization of simulations with the help of CoNM artifacts in the fourth experiment. The experiment clarifies the following: First, the ANN-based approximation of any kinds of system can be set up by the CoNM with the aid of the NMDL because the behavior of any system is imitated by the interplay of its approximating neurons. The approximation furthermore can realize an arbitrary accuracy, which is an interpretation of Hornik's universal approximation theory in a simulation context (Hornik, Stinchcombe, and White, 1989). Because of the approximation described, the individual system was considered to be part of a simulation system. Second, it was shown how to consider several ANN-based systems or rather sub-systems in one joint ANN-based simulation system. For building a joint ANN-based simulation systems, individual ANN systems are interwoven by neuronal connections furthermore. Third, it has become clear how knowledge carriers can be considered in simulations on a quantitative base, so that an organizational knowledge transfer model can be set up. Since each approximated modeling object is considered an ANN-based system, its processual manifestation is based on neuronal knowledge transfers. Due to these three key findings, neuronal simulations can be carried out scenario-wise, so that simulations can be based on data collection, on live data or on data generated by the simulation systems, which corresponds to a prediction. Further, the novel form of ANN-based simulation system stands as foundation for new kinds of ANN-based optimizations. Summarizing, this demonstration has achieved the following:

- ⇒ Individual systems have been set up as knowledge carriers.
- ⇒ The group of individual systems has been set up as knowledge carrier.
- ⇒ The neuronal knowledge base has been used for carrying out process simulations that are realized based on ANN.

The sixth demonstration (section 6.6) presents the optimization of ANN-based process systems on behalf of learning procedures. The fifth experiment presented two optimization attempts using a new kind of learning procedure that crystallizes knowledge about process parts in the sense of lower cognitive levels. The sixth experiment has presented two optimization attempts using traditional learning procedures within the novel context of ANN-based process systems. Since knowledge has not been crystallized, here, the fluent knowledge about process parts has been addressed by the sixth experiment. Both experiments showed the valid elimination of ineffective process parts as well as the preservation of effective process parts. Hence, a novel form of neuronal process optimization has been demonstrated. Summarizing, this demonstration has achieved the following:

- ⇒ A novel learning principle has lead to process optimizations. It has preserved crystallized knowledge of ANN.
- ⇒ Standard Backpropagation procedures have lead to process optimizations. These have focused on the fluent knowledge of ANN.
- ⇒ The neuronal knowledge base has been used for management purposes. Knowledge carriers have been identified to be removed if they are not relevant for the intended behavior. The removal successfully has improved the processes.

All in all, considering the key-findings of the demonstrations presented, the concept of neuronal modeling (CoNM) has been shown for the purpose of neuronal process modeling (NPM), neuronal process simulating (NPS) and neuronal process optimizing (NPO). Through this, the conceptual discussions of modeling in the *Process Domain*, in the *ANN Domain* and in the *ANN Process Domain* (cf. section 3.1.1) have been exemplified.



Evaluation

7

The evaluation considers the demonstration results presented in chapter 6 and examines to what extent the designs presented in chapter 4 have been proven in practice and whether the requirements of this contribution (see section 3.1.3) have been addressed successfully. In accordance with Peffers et al. (2007), this is the fifth step of a design-oriented research proceeding, and the section presents the CoNM evaluation of individual sub-artifacts following the methodological approach presented in section 3.3.

In general, an evaluation shall clarify the value and the benefit of an artifact after its professional assessment (Fettke and Loos, 2004a; Weber, 2015). As these attributes are dependent on the view of the corresponding stakeholder group (Fettke and Loos, 2004a; Österle et al., 2010; Mertens, 2010; Weber, 2015), the particular stakeholder information needs must be addressed by the evaluation (Weber, 2015) as the following describes.

- (A) The proceeding of an evaluation shall consider the following (Giel, 2013, p. 29):
1) the definition of the *evaluation purpose* and *evaluation objective*; 2) the identification of the information need for stakeholder groups and the formulation of an *evaluation question*; 3) the draft of an adequate *evaluation design*; 4) the *data collection* for carrying out the evaluation; 5) the analysis and interpretation of data collected so that the *potential* becomes transparent; 6) the *back referencing* to the initial evaluation question and evaluation objective. As this proceeding considers the stakeholder's information need in phases, target-oriented information addressing per stakeholder is supported. Individual phases, including

Supplementary Information The online version contains supplementary material, such as Appendix A, B, C and D whose references are indicated in the text, available at https://doi.org/10.1007/978-3-658-35999-7_7.

their activities, have been visualized in Fig. 7.1. These will serve as orientation for the evaluations presented in the following.

- (B) The choice of the evaluation method: The design-science-oriented research particularly demands for the clarification of an artifact's *usefulness* (Becker, 2010), *novelty* and *generality* (Hevner et al., 2004). However, their perception and the evaluation outcome depends on the choice of the *intended truthfulness understanding*; One can find four theories of truth that are meaningful for design-science-oriented research. Each demands for an individual understanding about truth and therefore leads to different evaluation results (Fischer, 2010; Wenturis, Hove, and Dreier, 1992, p. 122–146).

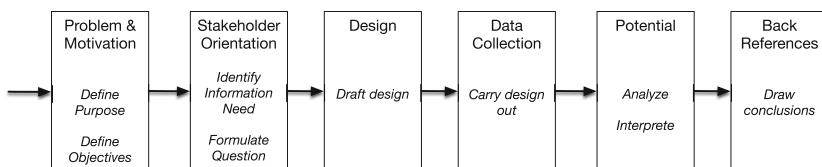


Figure 7.1 Methodological Approach for the Evaluation

- The *correspondence theory of truth* defines truth as compliance of a bearer of truth (e.g., a statement) and a counterpart (e.g., facts).
- The *coherence theory of truth* defines truth as the integration of a logical as well as comprehensible opinion with a logical as well as comprehensible system of opinions.
- The *consensus theory of truth* defines truth as the result of a professional, critical and ideal discourse.
- The “*formal truth*” defines truth by a valid proof.

The choice of *intended truthfulness understanding* and the corresponding theory of truth then leads to the *evaluation method* required (Weber, 2015, p. 331) and provides the base for the subjective reality of the stakeholders. Hence, evaluations considered in the following section will issue the method selection in harmony with the underlying truth understanding individually.

- (C) The choice of the intended quality field (Fettke and Loos, 2004a): As shown in Fig. 7.2, in general, the evaluation can address the construction process of an artifact or (not exclusively) the outcome of the artifact construction, which refers to the artifact itself. Further, the process or the outcome of the artifact demonstration can be issued.

The individual stakeholder interest can be connected to the desired quality field. For instance, the individual users and companies will mainly be interested in the quality of the demonstration process outcome, because it is attractive for product innovations. Politicians can further issue social changes. While trainers will mainly be interested in the question as to how the quality of the demonstration process can be improved, researchers will be willing to draw attention to the artifact constructed. The construction process, for instance, of the CoNM will be evaluated by the supervisors and evaluators of this contribution. For design-science-oriented research, as Peffers proposes and this contribution refers to, the demonstration of an artifact has been issued. Since the evaluation quality field of the *Demonstration Process Outcome* has been issued during the experiment-specific evaluations in chapter 6, the following will focus on the quality field of the *Construction Process Outcome* which issues the CoNM.

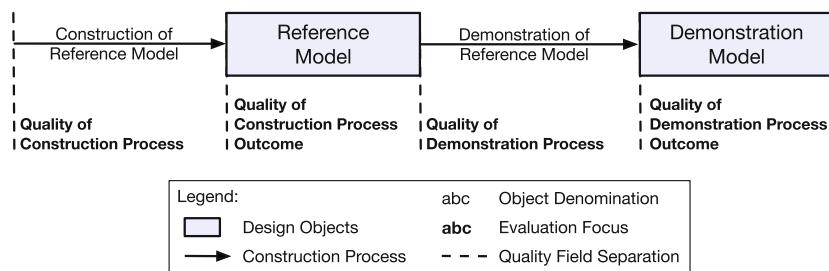


Figure 7.2 The Evaluation Quality Fields (following Fettke and Loos, 2004a)

7.1 Findings

As the individual evaluations of each sub-artifact have been presented in their corresponding demonstration sections, the following presents individual conclusions about each sub-artifact in the joint CoNM context. For this, the following subsections focus on an individual sub-artifact.

7.1.1 Selections on Behalf the CoNM-SESS

The sub-artifact presented here refers to the evaluation of the Generic Evaluation and Selection System (GESS), which was designed in section 4.2 and has been

demonstrated within the CoNM context in section 6.1. This concrete manifestation of the GESS is referred to as CoNM-SESS.

Problem and Motivation: A great number of concepts come from heterogeneous domains and are hard to compare because they satisfy individual intentions or differ in details. The GESS was constructed to deal with different kinds of challenges; apart from the complex characterization of OoIs, the trade-off of subjectivity and personal expertise as well as objectivity and broad competence was intended to be considered by this tool. The *purpose of the evaluation* is to examine the selection of candidates for an artifact construction. In this study, it will be examined whether the tool called GESS was able to deal with challenges mentioned and whether it has selected attractive candidates successfully.

The *evaluation objective* is to examine to what extent the GESS has been successfully applied to the CoNM context and selected OoIs are suitable for the CoNM construction. This issues the suitability of the GESS for the implementation in arbitrary contexts.

Stakeholder Orientation: Since the GESS is designed to be applied to a certain context, which was, as mentioned in section 6.1, the CoNM context, the parties interested in the evaluation primarily include researchers and companies that would like to identify the best foundation for an artifact construction. Using this, the GESS constructed gets evidence for the following: First, the GESS can be considered for the construction of next CoNM iterations having different OoIs, perspectives, aspects and test persons. Second, the GESS can be considered for the construction of different types of artifacts, for example, non-CoNM artifacts. Third, the GESS can be considered for the evolution of selection methods, for instance, for variants of criteria-based selection methods dealing with different aspect clusters, perspectives, test persons and OoIs.

Hence, the evaluation question is, “To what extend has the GESS progressed a best CoNM foundation?” Please note here that not the only best CoNM foundation has been issued because a different set of test persons, aspects, perspectives or OoIs could lead to alternative selections. Hence, the best selection rather represents a local optima than the global best solution.

Design: For the design of an evaluation instrument, the evaluation method must be chosen depending on the underlying understanding of truth. For the design-science-oriented proceeding, relevant understandings have been considered as follows:

First, the GESS evaluation design could consider formal proofs, as the GESS artifact is formally specified. Since these methods build on the specific selection of

OoIs and aspects listed in the CoNM-SESS sets, statements could not generalize over any artifact presented within the current body of knowledge. Hence, any statement is limited to the assumption of having listed the most attractive elements from the body of knowledge. These have been identified by the SLR and Bloom's taxonomy before (cf. the Systematic Selection Process at Fig. 4.5). So, methods from the "formal truth" understanding are not attractive.

Second, alternatively, focusing on the opinion of several evaluation experts, a *virtual discourse*, *focus groups*, *expert surveys* or *verbal protocol analyses*, could be chosen as the evaluation method (Fischer, 2010). As only one version of the CoNM-SESS is available, and an understanding from the viewpoint of the consensus theory of truth demands for higher numbers of test cases (Fischer, 2010; Weber, 2015, p. 331), consensus-based methods are dismissed here.

Since the CoNM-SESS has been demonstrated as an empirical instrument, one can also interpret individual OoIs to be test cases. Since these have been evaluated as part of the demonstration process (cf. demonstrations presented in section 6.1), these are not considered for the GESS evaluation design.

Third, methods supporting the correspondence theory of truth demand for a great number of observations, too. Examples can be found in the *laboratory experiment*, *field experiment*, *positivistic case study* or *field study* (Fischer, 2010). These have been dismissed for the same reasons.

Fourth, the evaluation design could consider methods supporting an understanding, which is in accordance with the coherence theory of truth. This is attractive because of the following reasons: The GESS leads to a characterization of OoIs considered, which is on behalf of numerical values only. The subsequent interpretation then derives qualitative insights. These can be considered logical and comprehensible opinions. They support the integration with the current body of knowledge because on the one hand, the body of knowledge can be seen as the comprehensible and logical system of opinions with whom they are integrated with. On the other hand, each aspect's and OoI's origin highlights the connection to the body of knowledge. Here, feature-based evaluation methods refer to the *SLR*, *meta-model-based evaluation*, *master reference model-based evaluation*, *theory-based evaluation* and *ontology-based evaluation* (Fischer, 2010).

The evaluation criteria refer to the combination of well-known features from the scientific discourse that have been proven in practice. By this, an objective evaluation is supported (Weber, 2015, p. 332). Therefore, features refer to the fulfillment of the qualitative criteria issuing the artifacts' *usefulness* (Becker, 2010), *novelty* and *generality* (Hevner et al., 2004).

Data Collection: The data collection considers the GESS only. Evaluations of individual OoIs have already been provided by the SLR and the theoretical foundation (chapter 2) as well as by the CoNM-SESS demonstrations (section 6.1). Hence, these are not issued again.

For the evaluation of the artifact, each criteria has been issued by argumentation, so that the following considers features individually.

Potential: Regarding the GESS design presented in section 4.2 and the demonstration as CoNM-SESS presented in section 6.1, the potential becomes transparent by three criteria.

a) Usefulness: The usefulness can be clarified by the following aspects:

- (1) With the aid of the GESS, any construction task can be operationalized so that the OoIs available for selection can be analyzed regarding the aspect fulfillment of the artifact to be constructed. Considering the Eq. 4.12, the selection aims to select OoIs that maximize the mean of criteria and minimize the variance. With the implementation of the GESS within the CoNM context, which refers to the CoNM-SESS construction, the GESS has been demonstrated successfully to be an empirical tool. Further, it has been shown to be usable for the artifact-specific adaptation.
- (2) With the aid of the CoNM-SESS, the concrete task of the CoNM construction has been operationalized so that the aspect fulfillment of an OoI can be considered in regard to the CoNM construction. Moreover, any tool or concept available in literature can be analyzed to be a CoNM foundation, and the CoNM-SESS demonstrates the successful use for selections for three times: first, as marginal case of one test person, the use by the author of this contribution, second, the use of experts that are able to evaluate an OoI or aspect cluster, and third, the use of experts that should be able to evaluate an OoI or aspect cluster.
- (3) Ranging OoIs and aspects on an equal footing, the GESS has provided a common base for the characterization of OoIs. As the comparison of different kinds of OoIs has been identified as challenge, the successful creation of a comparability for software tools, modeling concepts, biological mechanisms, etc. has been demonstrated by the CoNM-SESS. As it deals with an ordinal scale having numbers, the following becomes clear: first, the individual OoI as-is performance, which represents the current body of knowledge; second, the relative performance of an OoI when compared with another OoI; third, the relative performance of tools according to different

perspectives or aspect clusters; and fourth, a delta that can be derived for to-be tools so that a target-oriented development is enabled. With this, the tool clarifies its use to overcome evaluation challenges mentioned.

- (4) The GESS has demonstrated the possibility to generate plausible interpretations for the following three kinds of possibilities: first, the performance of individual OoIs regarding individual aspects and perspectives, second, the fulfillment of individual aspects or perspectives of all OoIs considered, third, the preferences of test person clusters such as experts and knowing experts. Since each of the three possibilities enables the selection of an OoI, the use of an evaluation focus for this selection has been demonstrated. By this, various challenges of an OoI selection have been addressed.
- (5) Beyond the context of the GESS demonstration presented, the CoNM has been built based on the foundation identified by the CoNM-SESS. Here, the use and the quality of selections by the GESS have been verified.

b) Generality: Since additional CoNM-relevant aspects and CoNM-relevant OoIs can be included in the CoNM-SESS easily, or irrelevant aspects and OoIs can be disregarded, results do not claim for completeness. So, the generality of the GESS is not limited. Instead, the selected set can serve as the first examination of the CoNM domain and stand as a reference point.

Since it was possible to generate plausible insights on behalf of the CoNM-SESS which are in harmony with insights generated by the SLR, an indicator for the generation of valid results becomes transparent. Furthermore, the CoNM-SESS has lead to more detailed results than the SLR or analyses on behalf of Bloom's taxonomy (Bloom et al., 1956) that are in harmony with the SLR. So, an indicator for the generalization of results can be identified by the CoNM-SESS demonstration. Since the CoNM has been built based on the foundation identified by the CoNM-SESS, the functioning of the GESS has been proven.

c) Novelty: The CoNM-SESS implemented the criteria from the CoNM meta-model (Fig. 4.4). Notably, a similar feature-based evaluation method has not been constructed before.

Further, a set of tools have been identified as CoNM foundation for the first time. The CoNM has been constructed based on these tools.

With the aid of the GESS visualization design, further, the *ANN Process Domain* has been made transparent for the first time. As not any OoI that satisfies this domain is available, the research gap has become clear by the colored visualization of the CoNM-SESS.

Back References: The *evaluation question* (“To what extend has the GESS progressed a best CoNM foundation?”) can be answered in regard to the criteria the GESS has been evaluated with; the GESS has progressed a CoNM foundation, since first, it has been proven to be useful in various dimensions for the selection of best OoIs, second, it has been proven to be novel in various aspects, and third, it has been proven to be able to stand as a generalizable tool.

Since a selection of candidates for artifact construction has been carried out and evaluated by the CoNM-SESS demonstration, and the tool constructed has been demonstrated to be able to select candidates regarding their criteria fulfillment of individual OoIs, one can say that attractive candidates can be identified by the GESS. The *evaluation purpose* so has been reflected.

The *evaluation objective* of an examination of the GESS has been reflected since challenges of a selection were addressed specifically. Here, it was confirmed that the GESS was able to deal with challenges as such, multiple persons, multiple criteria, multiple perspectives, as well as heterogeneous OoIs that are hard to compare. As the GESS has been applied successfully to the CoNM context and selected OoIs have been deemed suitable for the CoNM construction, an application to further contexts is attractive.

Regarding the requirements provided in section 3.1.3, the GESS supports the selection of concepts that fit to CoNM requirements. In the case of the CoNM-SESS, one can further find the implementation of the generic meta-model constructed. Hence, the GESS contributes as a sub-artifact of the CoNM at the time of construction of the main artifact of the CoNM.

7.1.2 Sub-Artifact-Specific Evaluation

The following will present an evaluation carried out on the level of the CoNM sub-artifacts. Beside the experiment-wise evaluation presented after each experiment in chapter 6, sub-artifacts have been examined in the CoNM context here. This has been realized before the joint modeling concept of CoNM is evaluated in section 7.1.3.

Problem and Motivation: Faced with a great number of CoNM artifacts created and requirements for the CoNM, the following examines the CoNM requirement fulfillment because of individual sub-artifacts. The *purpose of the evaluation* refers to the examination of the requirement fulfillment of the CoNM by its sub-artifacts. By this, it becomes clear as to which specific artifact contributes to the fulfillment of the joint set of requirements. The *evaluation objective* is to examine the requirement fulfillment of the joint CoNM. It will be examined as to what extent it has been built successfully as a modeling concept and whether it is suitable to carry out a neuronal process modeling.

Stakeholder Orientation: Since the CoNM is designed to be applied to the *ANN Process Domain* (section 3.1.1) and to access the NPM, NPS and NPO context, the parties interested in the evaluation primarily include researchers and companies that would like to identify the best manifestation of objects being approximated by the neuronal process modeling. By this, the CoNM constructed gets evidence for the following: First, the CoNM as a tool can be considered for the approximation as well as the construction of different types of objects such as for process networks, processes, tasks, activities, activations. Second, the CoNM can be considered for the evolution of trained ANN, such as for learning organizations, most complex ANN interpretations or the demonstration of the knowledge spiral on a neuronal level. Third, the CoNM can be considered for the construction of next CoNM iterations, such as for addressing different NEA configurations, having different modeling languages or tools supporting the CoNM proceeding.

Hence, the evaluation question can be put forth as, “To what extent has the CoNM manifested the best CoNM foundation?” Please note here that not the one and only best CoNM manifestation has been issued because a different set of evaluators and requirements could lead to alternative CoNM manifestations. Hence, the best selection rather represents a local optima than the global best solution.

Design: For the design of an evaluation instrument, the evaluation method must be chosen depending on the underlying understanding of truth. For the design-science-oriented proceeding, relevant understandings have been considered as follows:

First, the CoNM evaluation design could consider formal proofs. Since the CoNM sub-artifacts are not specified on the same level of formalization, some refer to mathematical models, some to implementations, some to experiments; the formal proof is not an option. So, methods from the “formal truth” understanding are not attractive.

Second, focusing on the opinion of several evaluation experts, a *virtual discourse*, *focus groups*, *expert surveys* or *verbal protocol analyses* can be chosen as the evaluation method (Fischer, 2010). As only one version of the CoNM is available and an understanding from the viewpoint of the consensus theory of truth is based on higher numbers of test cases (Fischer, 2010; Weber, 2015, p. 331), consensus-based methods have been dismissed here.

Third, methods supporting the correspondence theory of truth demand for a great number of observations, too. Examples can be found in the *laboratory experiment*, *field experiment*, *positivistic case study* or the *field study* (Fischer, 2010). These have been dismissed for the same reasons as above.

Fourth, the evaluation design could consider methods supporting an understanding, which is in accordance with the coherence theory of truth. This is attractive for

the following reasons: The CoNM leads to a requirement fulfillment by sub-artifacts considered, which is on behalf of possibly objective criteria only. The subsequent interpretation then derives qualitative insights. These can be considered logical and comprehensible opinions. They support the integration with the current body of knowledge; on the one hand, the body of knowledge can be seen as the comprehensible and logical system of opinions with whom they are integrated, and on the other hand, each requirement fulfillment of an sub-artifact highlights the connection to the body of knowledge. Here, feature-based evaluation methods refer to the *SLR, meta-model-based evaluation, master reference model-based evaluation, theory-based evaluation and ontology-based evaluation* (Fischer, 2010).

Since the requirement fulfillment needs to be verified, the evaluation criteria refer to the requirements as they have been presented in section 3.1.3. These have been interpreted from the scientific discourse since they have been discussed over the course of four years in different settings (cf. section 8.2.2). By this, an objective evaluation can be supported (Weber, 2015, p. 332).

Data collection: The data collection considers the requirement fulfillment only. Evaluations of individual sub-artifacts have been carried out by the author of this contribution. For the evaluation of the sub-artifacts, each criteria has been issued with argumentation so that the following considers requirements presented in section 3.1.3 individually.

Potential: As the CoNM sub-artifacts worked out, the potential becomes clear by the fulfillment of requirements identified. The following clustering and numbering refer to the clusters and requirement numbers of section 3.1.3. So, the linking of requirement and reasoning about the requirement fulfillment can be reproduced easily. Since section 3.1.3 has presented final versions of requirements that have been discussed with experts from the domain of knowledge management, process

Table 7.1 Requirement Fulfillment

Requirement		Description
Process Domain Side	Req. 1: ✓	An interpretation of process models to refer to and to consist of actual entities has been established so that neuronal structures can be considered while processes follow neuronal approximations. All has been considered during the specification of the modeling language called NMDL (section 4.6.4).
	Req. 2: ✓	Since the modeling items have been implemented to either stand as proper items or as reference of an item, the very same neuronal knowledge model can be integrated several times within a process model. So, it can be guaranteed that activations resulting from preceding tasks are kept, and that the neuronal network remembers its process history.
	Req. 3: ✓	Neuronal knowledge carriers have been prepared to be part of a simulation system. Hence, the use of an ANN's knowledge in process simulations is enabled.
	Req. 4: ✓	The modeled environmental factors (material such as non-material objects) have been integrated into prior mentioned ANN so that a common level of interpretation could be enabled.
	Req. 5: ✓	The outcomes (materialized such as non-materialized) of prior mentioned ANN have been considered in the process model and within process simulations. So, the effect of ANN-based knowledge transfers can be observed.

(Continued)

Table 7.1 (Continued)

ANN Domain Side	Req. 6: ✓	An interpretation of neurons as actual entities has been established so that neuronal tasks can be considered while neurons follow biological models. By this, both, the neuron's operational behavior and its learning processes, are harmonized. All has been considered at the specification of the modeling language called NMDL (section 4.6.4).
	Req. 7: ✓	The NMDL designed has considered the realization of parallel neuronal tasks that refer to ANN being activated simultaneously.
	Req. 8: ✓	The NMDL has considered tasks and activities that evolve over time by referring to recurrent neuronal networks.
	Req. 9: ✓	The NMDL has considered tasks and activities that are realized one after the other by the sequential activation of neuronal networks. Further, sequential tasks result from the sequential behavior of recurrent ANN.
	Req. 10: ✓	The NMDL has considered the control flow as it is realized with help of Boolean operators that are interpreted on the neuronal level.
	Req. 11: ✓	The NMDL has considered different levels of neuronal task abstractions so that items can refer to various levels of abstraction and granularity.
	Req. 12: ✓	The NMDL has considered sensory information and knowledge flows by individual knowledge items and attributes so that it becomes transparent as to how these are processed by the modeled neuronal networks.
	Req. 13: ✓	The NMDL has considered actuator information and knowledge by individual knowledge items and attributes so that it becomes transparent as to how these are processed by the modeled neuronal networks.
	Req. 14: ✓	A joint taxonomy (section 4.1) has been established that harmonizes individual terminologies of experts from different domains, wordings and contexts. A meta-model (Fig. 4.4) integrates relevant modeling objects.
	Req. 15: ✓	A modeling language has been designed (see NMDL at section 4.6.4) that can serve as a communication platform for different model constructors (machine-based as well as human) and different experts.
	Req. 16: ✓	A modeling concept (see CNPM at section 4.6) has been prepared, that is able to consider spatial dimensions. By this, a tempo-spatial behavior of the modeling space can be directly transferred to the target working space.
	Req. 17: ✓	A modeling concept (CNPM) has been prepared that can map the models constructed into the intended working space with the help of Augmented Reality techniques. Using this, behavior in the modeling space can be directly transferred to the target working space.
	Req. 18: ✓	Different approximation possibilities have been abstracted by a meta-model (Fig. 4.4). Further, model types have been systematized by the relation design of these kinds of systems (Fig. 4.14). Here, ANN systems are interpreted as models of original systems. Hence, model systems describing ANN systems with the aid of a modeling language indirectly describe the original systems. For the description of these systems, architectural levels have been introduced, such as the perspectives, views and hierarchical manifestations.
	Req. 19: ✓	Because of the following reasons, ANN systems follow modeling language-based models and vice versa: First, the relation of originals, ANN models and modeling language-based models have been systematized (Fig. 4.14); second, they have been functionally related. Using this, ANN systems can be easily constructed by modeling languages. Further, their current inner working can be visualized using the modeling language. Since models are stored by the file systems corresponding to the NEA (section 4.3), trained networks can be shared, reused and modified easily.
	Req. 20: ✓	By basing the CoNM tools on the Neuronal Enterprise Architecture (NEA) designed in section 4.3, input and target sources for ANN systems, such as third-party systems, external hardware and software systems and workflows can be integrated systematically and sustainably. Because of the NMDL's positioning as a communication platform, modeling results, analysis results, simulation results and optimization results are buffered by modeling items. So, any form of modification or neuronal activation becomes reproducible.
	Req. 21: ✓	A neuronal knowledge management model has been constructed (see NKMM at section 4.5) that enables the symbiotic management of human as well as machine-based knowledge carriers.
	Req. 22: ✓	A CoNM proceeding has been constructed (section 4.4) that considers the phase-specific project realization of CoNM projects. As each phase is accompanied with specific quality gates, the output generation can be controlled and the quality can be assured. Further, an NEA has been constructed (section 4.3) that considers different kinds of purpose-oriented model uses by separate management fields. So, a CoNM configuration can be implemented that supports the organization-wide and coherent model uses.
	Req. 23: ✓	ANN systems specified in section 4.6.1 systematically consider internal as well as external inputs and outputs. So, they can be considered to be neuronal enterprise model (corresponding to Fig. 4.14) and control external systems if this is intended.
	Req. 24: ✓	A selection method was designed (section 4.2) that supports the process of selection and evaluation of objects of investigations. It considers criteria from the process modeling (section 2.1), knowledge modeling (section 2.2), simulation modeling (section 2.3), process optimization (section 2.4) and corresponding tools (section 2.6).
	Req. 25: ✓	A procedure for the systematic extraction of knowledge from ANN systems has been designed (see SEKO at section 4.7). As this identifies knowledge within the ANN systems, the NKMM activity of <i>knowledge identification</i> is realized on a neuronal level by machines (cf. NKMM at section 4.5). Further, as identified knowledge is added to the corresponding models about ANN systems, mainly this refers to the views of (1) the functional perspectives, (2) the process ANN perspectives, (3) the knowledge perspectives and (4) the neuronal perspectives, a modeling in the sense of the NPM is exemplified. Based on the assumption to base knowledge on neuronal patterns and construct higher-order schemata by a combination of lower-order patterns (see knowledge construction axiom at section 1), the SEKO identified knowledge is integrated with the KM identified knowledge. Hence, with the aid of the NMDL (section 4.6.4), the organizational and neuronal knowledge base can be harmonized.
	Req. 26: ✓	A learning principle or rather training algorithm has been developed that unlocks ANN capacities at a certain system border by adding further neuronal sub-structures (see LPNR at section 4.8.1). Since activities have been complemented with tempo-spatial positions (see $O_i^j, P_j^l, Q_k^m, R_k^l$ at CNPM at section 4.6.1 and section 4.6.2), and adding of neuronal sub-structures can be stopped when adequate new capacities have been unleashed, the procedure has been compared with a recursion. Further, by the novel <i>NPO Backpropagation</i> (section 4.8.3), weight adjustments can be restricted to a spatial area or rather to a certain level of system interconnections. This enables the modification of spatial relations only.
	Req. 27: ✓	A learning principle or rather training algorithm has been developed that unlocks ANN capacities at a certain system level by respecting the pre-trained neuronal sub-structures (see <i>NPO Backpropagation</i> at section 4.8.3). While pre-trained ANN-based knowledge bases are crystallized, the building of more complex knowledge objects is enabled, and the construction of cognitive levels in the sense of an architecture is supported. Since weight adjustments are built during the consequent error flow throughout the ANN, the procedure has been compared with backpropagation.

(Continued)

Table 7.1 (Continued)

Implm. of Software Prototypes	Req. 28: ✓	The CoNM server has been implemented as a central computing structure for training runs. Experiments have demonstrated how training processes of neuronal networks can be realized by CoNM tools.
	Req. 29: ✓	The CoNM server has been implemented as a central computing structure for test runs. Experiments have demonstrated how testing processes of neuronal networks can be realized by CoNM tools.
	Req. 30: ✓	Since the CoNM server has been based on the NEA specified (section 4.3), through experiments, different kinds of warehouses and a central memory structure have been demonstrated. Further, the experiment in section 6.5.1 has demonstrated the integration of external production systems and the use of real-time data streams.
	Req. 31: ✓	Since the experiments have been initialized by the modeling tool called <i>Modelangelo</i> , the construction of neuronal networks has been realized and decentralized on ThinClients, such as laptops and desktop computers.
	Req. 32: ✓	The experiments have demonstrated how the coordination efforts of decentralized clients, the CoNM application server, the file server and the database server (cf. the architectural setup implemented at Fig. 5.2) can be coordinated in the background and comfortably via the Internet and WLAN structures. The experiment in section 6.3.1 has demonstrated the coordination in live modeling contexts. The experiment of section 6.5 has demonstrated the coordination in live simulation contexts, including third-party systems and workflows.
	Req. 33: ✓	The initialization of neuronal networks has been realized with the aid of the modeling tool called <i>Modelangelo</i> . Experiments have demonstrated how ANN can be constructed comfortably via drag and drop mechanisms.
	Req. 34: ✓	The integration of modeling tools and programming libraries are realized by the CoNM server implemented. For instance, the automatic network structure creation based on neuronal process models has been demonstrated by the experiments. This was realized by the background system; using CoNM mechanisms, more basic mechanisms of the programming library of <i>PyBrain</i> were called.
	Req. 35: ✓	The integration of simulation processes and programming libraries are realized by the CoNM server implemented. The start, stop and further control elements of neuronal process simulations have been provided by the modeling tool called <i>Modelangelo</i> . Using CoNM mechanisms, more basic mechanisms of the programming library of <i>PyBrain</i> were called. The CoNM server tool has demonstrated in experiments how the NPS is realized by a background system.
	Req. 36: ✓	The integration of optimization processes and programming libraries are realized by the CoNM server implemented. The start, stop and further control elements of neuronal process optimizations have been provided by the modeling tool called <i>Modelangelo</i> . Using CoNM mechanisms, more basic mechanisms of the programming library of <i>PyBrain</i> were called. The CoNM server tool has demonstrated in experiments how the NPO is realized by a background system.
	Req. 37: ✓	The experiments presented have referred to illustrative examples that shall visualize basic ideas of the CoNM. This has supported comprehensibility of the CoNM modeling concept.
Demonstration and Evaluation	Req. 38: ✓	The experiment shown in section 6.5 has demonstrated the decomposition of process networks as super-components down to neurons (top-down modeling). The experiment in section 6.4 has demonstrated the composition of knowledge objects as sub-components (bottom-up modeling). So, throughout the experiments, a hybrid modeling has been demonstrated, which has manifested from the NMDL's taxonomy views.
	Req. 39: ✓	Based on the knowledge construction axiom (Ax. 1), the meta-model (see Fig. 4.4) holds a comprehensive collection of knowledge carriers. By the experiments in section 6.5, the knowledge generation of a selection of different kinds of knowledge carriers has been approximated. Here, we can find the examples of machines, persons and CPS.
	Req. 40: ✓	Since the experiments were realized by following the CoNM proceeding shown in section 4.4, its phases have been demonstrated. For instance, the experiment in section 6.3.1 has focused on the specification, implementation, training and testing phases. The experiment in section 6.5 has focused on the neuronal simulation phase. The experiments in section 6.6 have focused on the neuronal optimization phase and have issued the development of to-be-concepts.
	Req. 41: ✓	Coming to the artifact of the NMDL, through the experiments, the interplay of atomic taxonomy views (Fig. 4.18), complex taxonomy views (Fig. 4.19) and flow-oriented views (Fig. 4.20), including their items and perspective-associations, have been demonstrated. It was assured to have demonstrated each view in at least one demonstration.
	Req. 42: ✓	Since software prototypes and CoNM tools are based on the NEA shown in section 4.3, experiments demonstrated the functional proof of the CoNM of the first three NEA levels: NPM (experiment of section 6.3.1), NPS (experiment of section 6.5) and NPO (experiments of section 6.6). Hence, the implementation presents one example for a CoNM configuration.
	Req. 43: ✓	Since the models of all experiments were specified by the NMDL and carried out by the CoNM tools, throughout the experiments, it was demonstrated how to cooperatively model by human as well as machine-based model constructors. Each kind of model constructor complemented the models. For instance, the first experiment in section 6.3.1 demonstrated how to set up neuronal networks by humans and complement their weight size and inhibitory character by machines. The experiments in section 6.4 demonstrated how to identify knowledge within neuronal structures computationally and interpret them using human model constructors.
	Req. 44: ✓	A measure-based management model has been designed that has been labeled with NKMM (section 4.5). This has provided an ordering system that allows the structuring of measures in the sense of a process-oriented, neuronal-based KM. The NPO experiments in section 6.6 have demonstrated how to <i>preserve</i> knowledge of pre-trained ANN. So, these were considered as crystallized systems. Further, it was demonstrated how to <i>clean up</i> knowledge by removing corresponding ANN structures.
	Req. 45: ✓	Since the experiments in section 6.5 and section 6.6 have addressed the different kinds of ANN systems, namely (1) network systems, (2) process systems, (3) activity systems, and (4) activation systems, these ANN systems have been demonstrated.

(Continued)

Table 7.1 (Continued)

	Req. 46: ✓	The novel CoNM algorithms have been demonstrated as follows: The SEKO (designed at section 4.7) has been demonstrated through the experiments in section 6.4. So, it became clear how different kinds of knowledge objects can be identified. The learning principles (designed in section 4.8) have been demonstrated throughout the experiments. Here, ordinary learning, or rather training procedures, result in a change of an activation or a set of activations. Further, they result in a modification of the ANN's knowledge base and behavioral dimension. NPS and NPO, ordinary learning or rather training procedures result in a joint optimization of the neuronal system. This disregards the current knowledge base of an ANN. The NPO learning procedures created respect the prior trained ANN and focus on the interconnecting weights of ANN systems, which can have a spatial or cognitive-architectural meaning. This has been demonstrated by the experiments in section 6.6.
--	------------	--

modeling and optimization, simulation and AI and they have been optimized over the course of three years, Tab. 7.1 addresses the requirement fulfillment of final versions only.

Back References: The *evaluation question*, that is, (“To what extend has the CoNM manifested a best CoNM foundation?”), can be answered in terms of the requirements the CoNM has been evaluated with. The individual sub-artifacts have progressed the main artifact of CoNM since these have been proven to satisfy requirements of the neuronal process modeling. Hence, the CoNM can be identified to be able to stand as a generalizable tool firstly satisfying the *ANN Process Domain*.

Since the sub-artifact constructions have been carried out as proper research processes, and they have jointly been evaluated by the CoNM experiments several times, and further, the CoNM tools constructed were able to function toward achieving their objectives, these can serve as a toolset carrying the CoNM. The *evaluation purpose* so has been reflected.

The *evaluation objective* of an examination of the CoNM has been reflected since the requirement fulfillment of an individual sub-artifact in the context of the creation of a CoNM modeling concept was addressed specifically. Collectively, the sub-artifacts were able to satisfy the entire collection of requirements. Regarding the requirements provided in section 3.1.3, one can identify that the CoNM is able to realize neuronal process models.

7.1.3 Evaluations on Behalf the CoNM-SESS

The evaluation presented here refers to the evaluation of the CoNM as the main artifact that consists of sub-artifacts (cf. mixed method design at section 3.2.6) and is compared with best OoIs identified in section 6.1.

Problem and Motivation: Having identified a CoNM foundation and the best candidates, the CoNM is stressed by a great number of concepts, heterogeneous domains and OoIs that are hard to compare because they satisfy individual intentions or differ in details.

The *purpose of the evaluation* so refers to the evaluation of OoIs on an equal footing with OoIs in terms of the task of construction of the CoNM. It shall be examined whether the main artifact called CoNM was able to satisfy the criteria issued and whether it is more attractive than best candidates considered, that is, the so called stat-of-the-art.

The *evaluation objective* is to examine to what extend it has successfully improved the CoNM context, and whether it is superior to the OoIs suitable for the CoNM construction (CoNM foundation). This helps the CoNM to overcome the OoI limitations so that the *ANN Process Domain* can be established successfully.

Stakeholder Orientation: Since the CoNM is designed to be applied in arbitrary contexts, the parties interested in the evaluation primarily include researchers and companies that would like to identify best candidates for a neuronal modeling. From this, the CoNM constructed gets evidence for the following: First, the CoNM can be considered for the construction of next CoNM iterations, including different concepts. Second, the CoNM as tool can be considered for the construction of different types of artifacts, including artifacts aside BPs.

Hence, the evaluation question is “To what extent has the CoNM progressed the ANN Process Domain and other domains?”. Please note here that the ANN Process Domain becomes real the moment it is accessed with the aid of the CoNM.

Design: The design of an evaluation instrument has been realized depending on the evaluation method and the underlying truth understanding. Design-science relevant methods and understandings have been considered as follows:

First, the evaluation design could be based on formal proofs, since the CoNM-SESS is formally specified. For the same reasons, the GESS has dismissed this kind of evaluation (see section 7.1.1); methods from the “formal truth” understanding are not attractive.

Second, the evaluation design could be based on the understanding of the consensus theory of trust. Here, methods supporting the opinion building process from a wide range of experts is attractive. For example, by a *virtual discourse, focus groups, expert surveys or verbal protocol analyses* (Fischer, 2010), the consensus could be identified. Hence, by principle, these are very attractive for the evaluation design, particularly because the CoNM-SESS has been demonstrated as an empirical instrument surveying experts. However, the CoNM constructed primarily intends

to stand as proof-of-concept, and the collection of test cases is rather small. This would require the highly effortful briefing of experts with CoNM concepts before a survey is conducted. So, consensus-based methods are not attractive for the CoNM evaluation on a broad base.

Third, for the same reason of having a small test case collection, test cases are considered to be facts that are in compliance with the body of knowledge (bearer of truth); methods from a correspondence theory of truth side are not attractive.

Fourth, regarding the understanding, which is in accordance with the coherence theory of truth, the evaluation design could consider feature-based evaluation methods such as the *SLR*, *meta-model-based evaluation*, *master reference model-based evaluation*, *theory-based evaluation* or *ontology-based evaluation* (Fischer, 2010). Following the same argumentation as the GESS evaluation shown in section 7.1.1, a CoNM evaluation and its comparison with further OoIs is considered to be a comprehensible and logical system of opinions that is integrated in the current body of knowledge by the CoNM-SESS. So, feature-based evaluation methods from the coherence theory of truth are attractive for the CoNM evaluation.

Since a joint evaluation of the CoNM is desired to be on an equal footing with any other OoI, only this guarantees an objective evaluation; the evaluation design builds on the GESS designed in section 4.2 and refers to the CoNM-SESS manifested in section 6.1. Having created a characterization of the current body of knowledge and best candidates by the CoNM-SESS demonstration, the following focuses on the comparison with the CoNM as main artifact.

Beside the evaluation criteria coming from the CoNM meta-model and specific set of tools, analyses are oriented to well-known features from the scientific discourse that have been proven in practice. Namely, these features refer to the fulfillment of the qualitative criteria issuing the artifacts *usefulness* (Becker, 2010), its *novelty* and *generality* (Hevner et al., 2004). By this, an objective evaluation is supported (Weber, 2015, p. 332).

Data Collection: The data collection considers the instrument of the CoNM-SESS for the evaluation of the CoNM. Since the CoNM needs to be considered in comparison with contemporary objects of investigations (OoIs), Fig. 7.3 presents the overview from the expert base presented in section 6.1. Further, the CoNM evaluation is presented from the perspective of the author of this contribution and the CoNM evaluation from the expert base. The latter consisted of the ten experts that have been educated in CoNM concepts for three years. The concrete description of the figure is issued in the following.

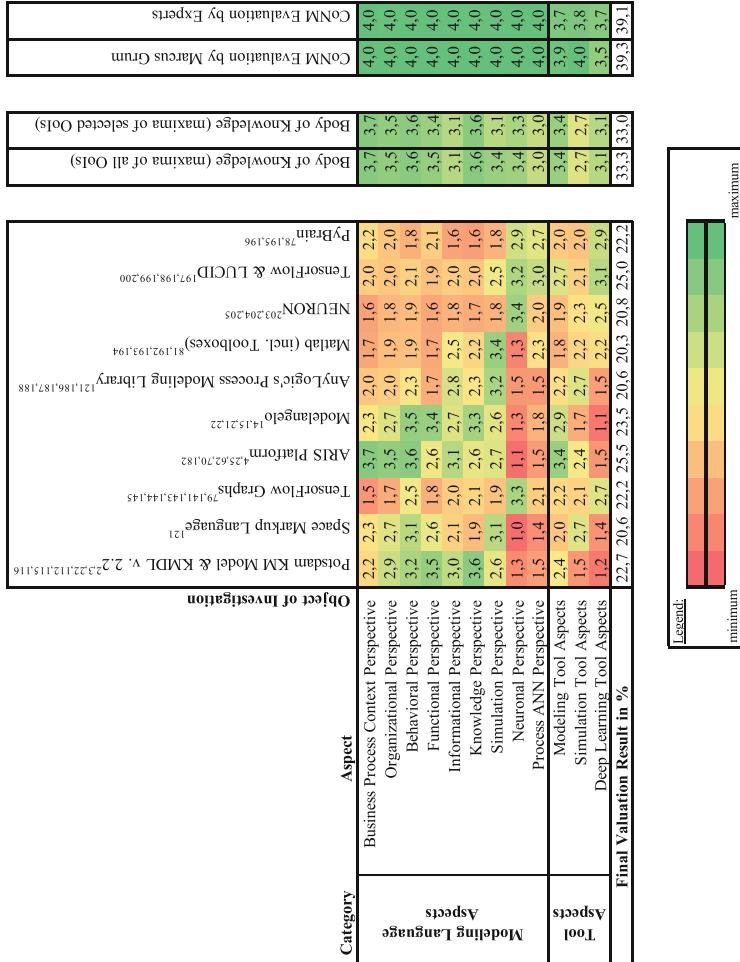


Figure 7.3 The Evaluation of the CoNM by the CoNM-SESS (Focusing the Mean)

Right within the center of the figure, one can see the mean values $V_{i,k}^{\mu,+}$ (Eq. 4.10) and the aggregated mean values $R_k^{\mu,+}$ (Eq. 4.11) of section 6.1. As further OoI, the “Body of Knowledge” has been presented. This is based on the maxima of mean values over all OoIs as a first variant. A second variant of the “Body of Knowledge” is based on maxima of values over the OoIs, which have been selected as foundation for the CoNM construction. The evaluation of the CoNM is presented on the right so that an aspect-wise, cluster-wise and OoI-wise comparison is enabled and the construction task can be issued according to Eq. 4.12.

In direct comparison with all the OoIs and the variants of the body of knowledge with the CoNM constructed, one can recognize the following:

1. The CoNM can be identified as superior in all the clusters identified. This means the following: First, the CoNM provides the most relevant modeling shapes required for a neuronal modeling (*modeling language aspects*). Second, the CoNM supports the neuronal modeling best because of its function as a modeling tool, simulation tool and deep learning tool (*tool aspects*). Hence, the CoNM has progressed the state-of-the-art in any cluster dimension.
2. The CoNM can be identified as superior in all the final valuation results. This means that the CoNM best addresses the joint aspect fulfillment of the relevant categories of first *modeling language aspects* and second *tool aspects*. Hence, the CoNM can be identified as the new state-of-the-art in the construction task addressed.
3. The CoNM satisfies the entire set of *modeling language aspect* clusters as well as their joint sets of individual aspects around a level of 100%. This is because the generic meta-model constructed in section 4.1 has been implemented in the CoNM without exceptions. Hence, the CoNM has realized a nearly perfect implementation of the modeling language aspects, which is, in the construction task context, addressed.
4. Slight deviations from a best aspect fulfillment can be identified in the category of *modeling tool aspects*. This deviation is because routines for a quality management have not been implemented by CoNM software prototypes, yet.
5. Slight deviations from a best aspect fulfillment can be identified at the category of *Deep Learning tool aspects* as well. This deviation is because the efficient data processing within multi-processor edge-computing environments has not been considered in CoNM software and hardware implementations. This refers to the speed optimal processing of ANN-based calculations. Further, the feature visualization has not been integrated with a first version of CoNM tools.

6. The CoNM evaluations of the expert base do not differ much from the CoNM evaluation of the author of this research.

Since the CoNM evaluation has been carried out by a great number of experts, the variance has been issued in Fig. 7.4, which presents the variance overview from the expert base presented in section 6.1, the CoNM evaluation from the personal expertise of the author of this contribution and the CoNM evaluation from the expert base mentioned above.

Right within the center of the figure, one can see mean values $V_{i,k}^{\sigma^2,+}$ (Eq. 4.10) and aggregated mean values $R_k^{\sigma^2,+}$ (Eq. 4.11) in section 6.1. As further OoI, the “Body of Knowledge” has been presented. This is based on the maxima as well as the minimum of variance values over all OoIs as a first variant. A second variant of the “Body of Knowledge” is based on maxima and minimum of values over the OoIs, which have been selected as foundation for the CoNM construction. The evaluation of the CoNM is presented on the right so that an aspect-wise, cluster-wise and OoI-wise comparison is enabled, and the construction task can be issued according to Eq. 4.12.

In direct comparison with all the OoIs and the variants of the body of knowledge with the CoNM constructed, one can recognize the following:

1. The CoNM can be identified as superior in all the clusters identified because of the smallest variances. This means the following: First, the CoNM provides the most common view on relevant modeling shapes required for a neuronal modeling (*modeling language aspects*). Second, the CoNM provides the most common view on its function as modeling tool, simulation tool and deep learning tool (*tool aspects*). Hence, the CoNM has increased the agreement about the state-of-the-art in any cluster dimension.
2. The CoNM can be identified as superior in all final valuation results because of the smallest variances. This means that the CoNM has lead to the most common view on the joint aspect fulfillment of the relevant categories of first *modeling language aspects* and second *tool aspects*. Hence, the CoNM has increased the agreement about the construction task addressed.
3. The CoNM satisfies the entire set of *modeling language aspect* clusters as well as their joint sets of individual aspects around a variance level of 0.01%. This is because the generic meta-model constructed in section 4.1 has been implemented in the CoNM without exceptions, which was recognized by the expert base. Hence, the CoNM has realized a perceived perfect implementation of the modeling language aspects, which is, in the construction task context, addressed.

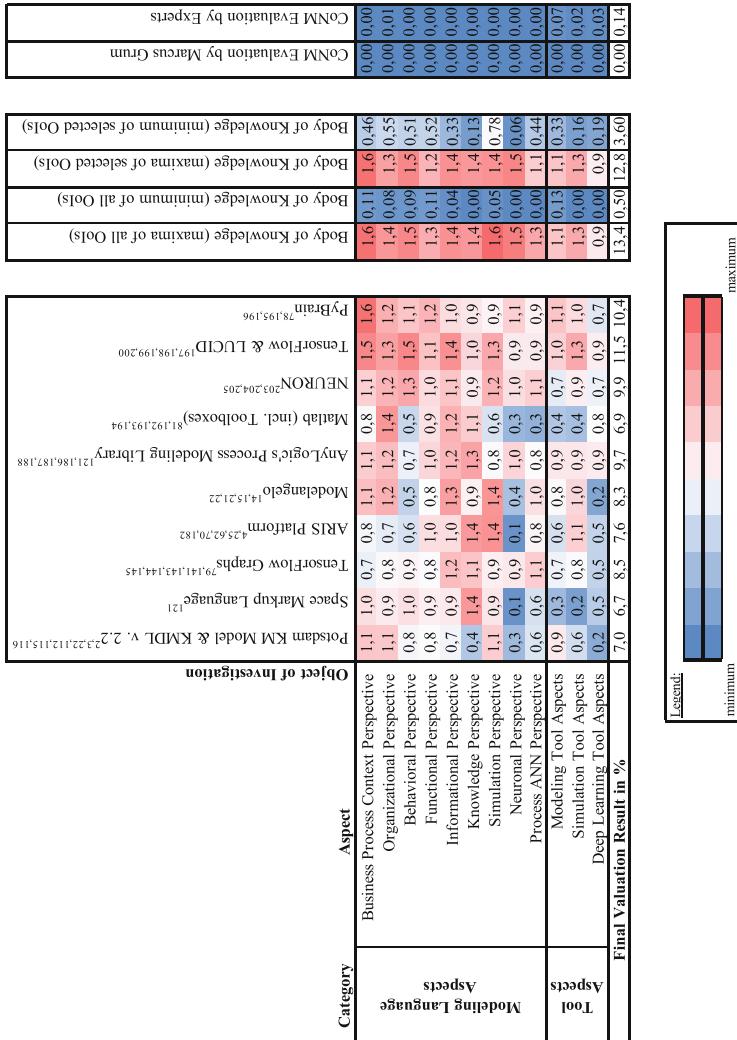


Figure 7.4 The Evaluation of the CoNM by the CoNM-SESS (Focusing the Variance)

4. Slight deviations from a best variance level can be identified in the category of *tool aspects*. This deviation is because implementations by CoNM software prototypes have lead to variances in the perceived aspect fulfillment. Although the CoNM does not reach the variance level of the joint body of knowledge, here, the best variance levels from the selected OoIs have been improved without exception.
5. The CoNM evaluations of the expert base do not differ much from the CoNM evaluation of the author of this research.

Taking insights of mean levels and variance levels together, the CoNM can be identified as superior in all the aspects and has created the best agreement on aspects evaluated. Hence, up to now, it satisfies the construction task according to Eq. 4.12 best. Later CoNM versions nevertheless still have a potential for improvement.

Potential: Regarding the comparison of best tools available and the CoNM constructed, which is based on the CoNM-SESS, the CoNM potential becomes transparent by three criteria in addition.

a) Usefulness: The usefulness can be clarified by the following aspects:

- (1) Since any construction task can be operationalized with the aid of the GESS, and the GESS has been implemented within the CoNM context, the CoNM main artifact has been evaluated with regard to the construction task of Eq. 4.12.
- (2) The CoNM has demonstrated to be the new state-of-the-art with regard to aspects, aspect cluster, and the joint set of aspects. This is based on both the mean of the evaluation of the expert base as well as the variance of the evaluation of the expert base. Further, the evaluation of the author of this contribution confirms the expert base.
- (3) Since the CoNM and OoIs have been evaluated on an equal footing, the GESS has provided a common base for the characterization of OoIs. So, the successful creation of a comparability for software tools, modeling concepts, biological mechanisms, etc. has been evaluated by facing the CoNM.
- (4) Beyond the benchmark on the base of the GESS presented, the CoNM has been identified as superior OoI. It therefore can stand as the new foundation for next CoNM versions.

b) Generality: Having considered the selected set of criteria (see section 6.1), the evaluation claims for generality with regard to the first examination of the CoNM domain. The CoNM constructed therefore can stand as the first reference point.

c) Novelty: With the aid of the GESS visualization design, the ANN Process Domain has been made transparent for the first time. As no OoI has been available, so far, the research gap became clear by the use of the GESS. Faced with the new evaluation results of the CoNM constructed, it became clear that this research gap has been addressed successfully.

Back References: The *evaluation question*, that is, (“To what extend has the CoNM progressed the ANN Process Domain and other domains?”), can be answered with regard to the criteria the GESS has been based on.

Because of the following three kinds of evaluations, which have been carried out with the aid of the CoNM-SESS, the CoNM tool has proven to be superior in a benchmark: first, the evaluation of candidates of a CoNM foundation, second, the evaluation of the body of knowledge as well as third, the evaluation of the CoNM constructed. Therefore, an attractive candidate can be identified in the CoNM tool. The *evaluation purpose* so has been reflected.

The *evaluation objective* of an examination of the improvement of the CoNM context has been reflected since challenges of a concept of neuronal modeling were addressed specifically. Here, it was confirmed that the CoNM was able to deal with challenges as such, separated domains, lack in the transparent dealing with and interpretation of ANN, as well as the identification of tacit knowledge. As the CoNM has been applied successfully to BP contexts and selected OoIs have been deemed suitable for the CoNM construction, an application to further contexts is attractive. The exact extent of improvement or superiority can be derived by the CoNM-SESS aspects.

Regarding the requirements provided in section 3.1.3, the CoNM supports the integration of concepts that fit to CoNM requirements. Hence, the CoNM contributes as a joint artifact to the construction of a concept of neuronal modeling.

7.2 Evaluation of the Practical Applicability

The evaluation presented here issues the practical applicability of the CoNM as the main artifact. Since the CoNM is applied to a well-defined, practical and real environment, the following can be interpreted as a mixed method design (cf. section 3.2.6) obtaining an empirical component.

Problem and Motivation: Having identified the CoNM as the main artifact, the CoNM is stressed by a real simulator providing different physical machines, real sensors, time-dependent production processes and disturbing factors. It shall be examined if the CoNM can capture problems and issues of a real system.

The *purpose of the evaluation* so refers to the evaluation of the CoNM's practical applicability. More concrete, it refers to the benchmark of the NPS with the real system and focuses on the examination whether the CoNM is able to stand as simulation system for a real setting. Furthermore, it refers to the benchmark of the NPO with the real system and focuses on the examination whether the CoNM is able to optimize processes of a real system.

The *evaluation objective* is to examine to what extent the CoNM is able to successfully simulate and improve a real system, and whether it is superior to the standard simulation system environment.

Stakeholder Orientation: Since the CoNM is designed to be applied in arbitrary contexts, the parties interested in the evaluation primarily refer to researchers and practitioners. Probably, the latter kind of interests come from small, medium and large companies that would like to identify issues from a practical application of a neuronal modeling. Through this, the CoNM constructed gets evidence for the following: First, the CoNM can be considered for modeling neuronal processes from real systems having different machines and process participants. Second, the CoNM as a tool can be considered for the controlling of different kinds of systems, for instance, for standard BPs or industrial processes. Third, the CoNM can be used as a tool for automatically optimized as-is processes of a system on behalf of the data collected from this system.

Hence, the evaluation question is “To what extent can the CoNM be applied to a real system?” It will also be examined as to how far one has to compromise.

Design: The definition of an environment for enabling the evaluation of the practical applicability of the CoNM has been well defined and structured by the following alphabetically arranged categories.

a) Mechanical Construction: The simulator has been built on behalf of the product called *Lego Mindstorms Education Ev3 Set*. Originally, this product was made by LEGO in order to enable school teaching in the higher classes of mathematics, physics, computer science and techniques (LEGO Group, 2013b). First, the mechanical construction of one robot arm (LEGO Group, 2013c) and three color sorter machines (LEGO Group, 2013a) were realized, which has been in accordance with the original building instructions of LEGO. Fig. 7.5 presents the construction outcomes when the corresponding instructions were realized carefully.

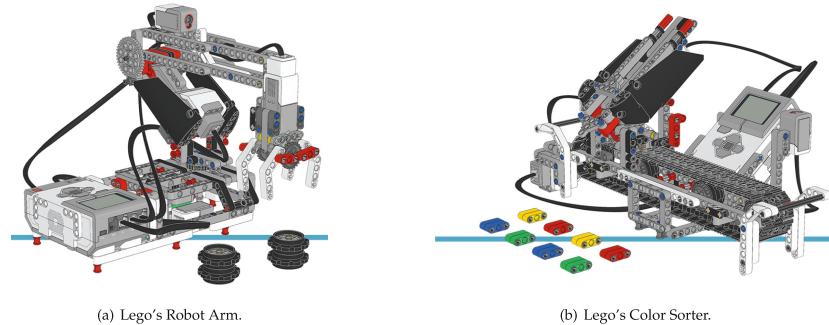


Figure 7.5 Lego Mindstorms EV3 Machines (original)

While Lego's robot arm can pick workpieces from and transport workpieces to three locations, which are on its left, its right and its middle position, Lego's color sorter moves a placing unit along the conveying belt it has been stuck to. Since it moves between four positions, it is able to drop a workpiece on these positions in dependence on the workpiece's color. For instance, the workpieces can be collected by four glasses standing right in front of the conveying belt, once for four different kinds of colors. The color has been determined by showing the workpiece manually to the color sensor attached to the device's right and storing it at the placing unit manually thereafter.

Second, the four kinds of compartments have been modified and placed side by side so that they are able interact efficiently. Here, a focus was set on a possibly high number of different kinds of interactions. This has lead to the following design decisions:

- The *robot arm* is placed in the physical center of the simulated production space so that it is able to pick workpieces from its two sides and it further can be accessed by its front or rather middle side for maintenance.
- The placing unit was detached from the conveying belt so that each conveying compartment can stand individually *transport machine*. Further, the touch sensor was removed because it was not required to identify the placing unit's position any more.
- The original placing unit was equipped with machine feet so that it can stand individually *production machine*. Since it is used flexibly in regard with the underlying production scenario, the color sensor was removed.

- The constructed production machines were positioned next to the transport machines so that they can drop products directly on the conveying belt of a transport machine.
- The *robot arm* is put on a plateau so that it is able to pick workpieces directly from the conveyors of its surrounding transport machines.
- One conveyor was put in a cellar so that its conveying belt represents the floor. Having put the statue of a human production worker on the conveying belt, the person was simulated to move to the robot arm so that it can realize a maintenance. Hence, it is referred to as *floor machine* from here on.

Fig. 7.6 presents the final mechanical arrangement of the four machines constructed. Since it is based on a collection of different autonomous machines, it is able to represent an industrial production system.



Figure 7.6 The Physical Arrangement of the Lego Simulator Units

The modified LEGO components have been fixed to a ground plate. This same plate has an opening for one of the transport machines so that the movement of either the maintenance worker or the cleaning worker can be simulated by simply putting

them on the machine's conveyor belt. The processing units for the corresponding machines have been positioned as a group of computers like a server farm at the front right-hand corner so that the communication of the machines can be read from the displays comfortably. Cables have been crafted from the scratch so that they can be guided through holes of the plate. Further, within the pedestal created, the power cables have been positioned. Human-like dolls have been taken from Duplo products because of their more realistic size. Through this arrangement, a tidy and modular appearance was achieved and the joint simulator could be carried from presentation to presentation.

b) Sensor Design: Abstracting the final arrangement of Fig. 7.6, Fig. 7.7 presents the ground plan of the final mechanic production space and the sensor positioning of the four CPS.

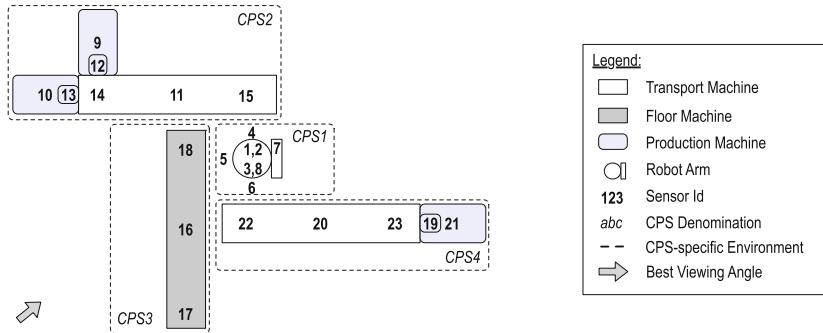


Figure 7.7 The Ground Plan of the Production Space and Sensor Positioning

The figure shows the positioning of 23 sensors that have been associated with four different kinds of physical, industrial-similar systems. Since each industrial system has been conceptualized to have 1) an individual sensing of its environment, 2) an individual motor set as actuators manipulating its environment, 3) an individual processor to process sensor data and identify decisions and 4) its individual communication channel via the Internet to *speak* with further systems, these four kinds of systems can be considered to be cyber-physical systems each (CPS), and they jointly manifest as cyber-physical production system (Gronau, Grum, and Bender, 2016). The concrete sensor specification can be found in Tab. 7.2. Since the *Sensor ID* presented here corresponds to the enumerating of Fig. 7.7, the sensors

Table 7.2 Lego Sensor Design

Sensor ID	System	Sensor Data	Component	Scale	Type
1	CPS1	Motor A	Gripper	Angle	physical
2		Motor B	Elbow	Angle	physical
3		Motor C	Base	Angle	physical
4		Workpiece detection right	Surrounding right	Boolean	virtual
5		Workpiece detection middle	Surrounding middle	Boolean	virtual
6		Workpiece detection left	Surrounding left	Boolean	virtual
7		Base switch	Base border position	Touch	physical
8		Elbow sensor	Elbow border position	Color	physical
9	CPS2	Motor A	Feeder 1	Angle	physical
10		Motor B	Feeder 2	Angle	physical
11		Motor D	Conveyor	Angle	physical
12		Material availability	Feeder 1	Boolean	physical
13		Material availability	Feeder 2	Boolean	virtual
14		Workpiece left	Conveyor	Boolean	virtual
15		Workpiece right	Conveyor	Boolean	virtual
16	CPS3	Motor D	Conveyor	Angle	physical
17		Person far	Conveyor	Boolean	virtual
18		Person close	Conveyor	Boolean	virtual
19	CPS4	Motor A	Feeder 1	Angle	physical
20		Motor D	Conveyor	Angle	physical
21		Material availability	Feeder 1	Boolean	virtual
22		Workpiece left	Conveyor	Boolean	virtual
23		Workpiece right	Conveyor	Boolean	virtual

can be associated with the corresponding positions within the spatial arrangement easily.

Focusing on the sensor types presented in the table, one can recognize about half of the sensors to be virtual. Although these could have been realized by physical sensors instead, they have been handled as virtual sensors per design decision because the simulator was intended to run for a conditioned simulation. This means that the simulator, for instance, runs until a certain number of products were realized successfully. Since the number of scenarios realized is unknown in advance, and the workpiece flow through the CPPS would stop when a physical sensor does not detect a physical workpiece to be available for the next production step, physical workpieces would have been needed to be refilled without excuse. With the aid of a virtual storage of workpieces, an endless number of workpieces can be assumed. Hence, the simulated CPPS does not stop because it has run out of workpieces.

c) Scenario Design: The simulator was designed to realize eight kinds of scenarios. These manifest because of three kinds of variables. First, the presence of a workpiece which needs to be transformed through a predefined sequence of production activities. Second, the attempt to realize maintenance by a production worker at the robot arm. Third, the presence of a cleaning activity that unintentionally can be perceived as maintenance activity by the corresponding sensors. Building on these random variables, scenarios can be characterized in accordance with the bivalent Truth Table of Tab. 7.3.

Table 7.3 Lego Scenario Collection

Cleaning	Workpiece	Maintenance	ID	Title
False	False	False	1	Idling the production
		True	2	Maintaining the robot
	True	False	3	Producing the workpiece
		True	4	Production and maintenance in conflict
True	False	False	5	Cleaning causes maintenance program
		True	6	Undisturbed cleaning and maintenance
	True	False	7	False alarm causes production stop
		True	8	Undisturbed cleaning and alarm situation

In accordance with the scenario IDs presented in Tab. 7.3, the following characterizes the four standard scenarios. Please note that these are not affected by the cleaning operation.

1. **Idling the production:** The production system pauses for the production slot of five moments. Here, none of the production machines, conveyors or robot arms is supposed to do anything.
2. **Maintaining the robot:** The production system does not produce an output. The robot arm activity is realized in order to verify the correct operation of its components by a maintenance worker. The scenario is realized by the five moments as follows:
 - 1) The maintenance is announced, which is simulated by CPS3.
 - 2) The maintenance worker approaches the robot arm.
 - 3) The robot arm is maintained. For this, CPS1 realizes the typical transport activity.
 - 4) The correct interplay of the robot arm and the transport activity of CPS4 is verified. A sound of CPS3 indicates the successful maintenance by the production worker.
 - 5) The production worker leaves the maintenance position.
3. **Producing the workpiece:** The production system realizes a number of pre-defined production activities in order to transform a workpiece at the corresponding machines, and an output is created at CPS4. Material input is provided at CPS2, and the production output can be picked up from the right of the conveyor belt of CPS4. The scenario is realized by the five moments as follows:
 - 1) CPS1 activates both its feeders.
 - 2) CPS1 completes its activation by the transport of the workpiece from its left to its right of the conveyor belt of its transport machine.
 - 3) The robot arm is activated in order to transport the workpiece from its right (conveyor belt of CPS2) to its left (conveyor belt of CPS4).
 - 4) CPS4 transports the workpiece to its right.
 - 5) CPS4 completes its activation by the activation of its feeder so that the production outcome is generated.
4. **Production and maintenance in conflict:** The robot arm is required by two processes: first, the production process that intends to transport a workpiece from CPS2 to CPS4 (see scenario 3); and second, the maintenance process that

intends to verify the operation of the robot arm (see scenario 2). In order to avoid an accident, the robot arm is stopped by an incident, and an alarm indicates the re-initialization of the production systems.

- 1) CPS1 activates both its feeders. Further, the maintenance worker announces its maintenance.
- 2) CPS1 completes its activation by the transport of the workpiece from its left to its right of the conveyor belt. CPS3 completes its announcement by approaching the maintenance worker at the robot arm.
- 3) The robot arm is stopped by the emergency routine in order to avoid an accident.
- 4) The alarm indicates the emergency case and the production stop. Regrettably, the production outcomes are destroyed because of this.
- 5) The four CPS reinitialize so that they are ready for the next production slot. The successful initialization is indicated by the typical initialization sound.

In accordance with the scenario IDs presented in Tab. 7.3, the following characterizes the four scenarios that manifest because of the cleaning operation.

5. **Cleaning causes maintenance program:** Originally, scenario 1 is realized. Regrettably, the sensor indicating the maintenance is activated falsely by the cleaning machine so that it seems as if a maintenance is realized (scenario 2). Here, unnecessary costs are generated because of incorrect activation of the production systems.
6. **Undisturbed cleaning and maintenance:** Originally, scenario 2 is realized. The sensor indicating the maintenance is activated falsely by the cleaning machine. Luckily, since the maintenance is realized simultaneously and the maintenance worker activates the sensor in addition, the false activation is not discovered.
7. **False alarm causes production stop:** Originally, scenario 3 is realized. Regrettably, the sensor indicating the maintenance is activated falsely by the cleaning machine so that it seems as if a maintenance is realized in addition (scenario 4). Hence, the production system is stopped unnecessarily, so unnecessary costs are generated by the false alarm situation. Further, no production output is generated.
8. **Undisturbed cleaning and alarm situation:** Originally, scenario 4 is realized. The sensor indicating the maintenance is activated falsely by the cleaning machine. Luckily, since the maintenance worker simultaneously activates the sensor, the false activation is not discovered and the alarm situation is caused correctly.

d) Software Design: In order to unlock a flexible programming of the EV3 products of LEGO, its original operating systems were replaced with *ev3dev*. It is a Debian Linux-based operating system that runs on various devices such as the *Raspberry Pi* or the *Beaglebone* and is provided under the GNU General Public License vers. 2.0 (Ev3Org, 2020a). Among the great collection of the *ev3dev*-supported programming languages, the Python-based *pybricks-micropython* was chosen for the EV3 programming because it provides basic motor routines and drivers for the hubs, motors and sensors. These routines enable the accurate controlling of the EV3 devices (Ev3Org, 2020b).

Fig. 7.8 presents the architecture of the Lego simulator constructed. It has been presented as UML's deployment diagram and intends to clarify the architecture of the Lego simulator as well as its integration with the architectural setup of demonstrators, which has been presented in Fig. 5.3.

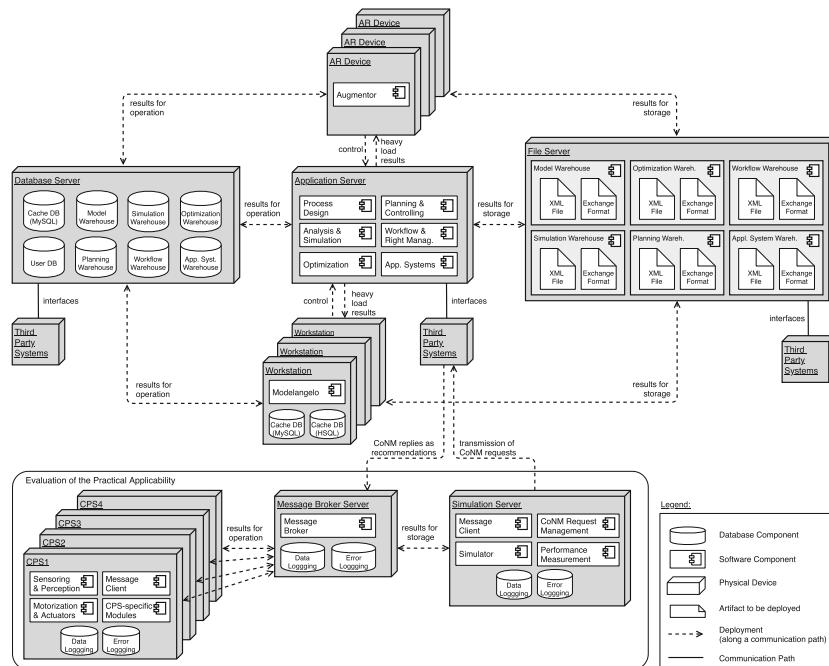


Figure 7.8 The Lego Simulator Architecture and CoNM Integration (as UML Deployment Diagram Variant)

In the figure, one can see that the CoNM is integrated by the simulation server. It transmits CoNM requests containing relevant simulation data and receives recommendations from the CoNM via the message broker. Please note here that the CoNM results could have been sent to the simulation server directly. Here, it was essential to run the messaging client and the CoNM requesting in multiple threads so that the simulation system did not get blocked because of effortful, time-consuming ANN calculations. Hence, asynchronously, the CoNM results were sent as a message to the simulation server when the CoNM processing has finished successfully. Finally, the CoNM requests were transferred directly to the corresponding CPS, so that these CPS could be controlled by neuronal instructions.

For the message broker presented in the figure, the *mosquitto vers. 1.6.11* server was chosen because of the following reasons (Light, 2017; Light, 2020): 1) It is an open source (EPL/EDL licensed) message broker. 2) It promises to be lightweight, so it is suitable for low power single board computers as well as full servers. This supports the reproducibility of simulations presented because any kinds of systems can be connected to the CoNM. 3) It provides the MQTT protocol vers. 5.0, 3.1.1 and 3.1, which implements a lightweight publish/subscribe model. Setting the focus on messaging of Internet of Things environments or rather Industrie 4.0 production systems, the messaging for low power sensors, mobile devices and embedded computers is supported.

The individual devices were conceptualized to realize an individual sensing and perception as well as an individual motorization and controlling of actuators autonomously with the aid of CPS-specific modules. Although each device logged simulation data and errors individually, relevant outcomes were communicated via the message broker to the simulation server. Here, the performance of the joint CPPS was evaluated. In order to keep the messaging effective, the implementation was designed to be satisfied by the following kinds of communication channels:

- Four *device-specific channels* have focused on the management of each device individually. For instance, via this channel, the device-specific initialization and configuration was instructed.
- One *simulator-specific channel* has focused on the joint and simultaneous controlling of the joint simulation system. For instance, the start of the individual simulation on each device was instructed via this channel.
- One *state-specific channel* has focused on the state-specific information, which might affect the joint simulation system. For instance, via this channel, the update of sensor data of neighboring systems was arranged, or the introduction of a system as messaging client was carried out.

- One *CoNM-specific channel* has focused on the transmission of CoNM recommendations. For instance, by communicating via this channel, the individual devices or the simulation server can be overruled by the CoNM recommendations.

Fig. 7.9 presents the communication sequence of the Lego simulator constructed. It has been presented as UML's sequence diagram and intends to clarify the communication of the Lego simulator and its integration with the NEA software that has been presented in Fig. 5.8.

Following a variant of the UML sequence diagram type, in the figure, one can see the system interactions in time sequence. From this, it becomes clear as to which software and hardware components are required by a certain use case. While hardware and software components ("systems") have been visualized by UML's component shape and refer to the CoNM's application server, simulation server, message broker and message client, each component's lifeline has been visualized by a dashed line. Since different kinds of interactions, the so called message exchanges or operation calls, have been visualized by horizontal arrows, the course of time becomes transparent by a vertical time axis from the top to the bottom. While synchronous interactions block the corresponding lifeline during the call processing, this is visualized by the gray activation box between two synchronous interactions; asynchronous interactions do not block a system's lifeline. Although principally UML's sequence diagram type visualizes the flow through a decision tree of one individual system call, as a variant, the figure shows the flow through decision trees for the collection of use cases. Hence, as a variant, use cases have been complemented by UML's well-known note symbol. Since use cases have been arranged depending on their composition on the left of the figure and the corresponding sequences have been associated by horizontal gray lines, the collection of use cases represents the simulation procedure as follows:

1. The clients are started. They individually introduce themselves at the message broker.
2. The simulator is started. It introduces itself at the message broker. Then, it instructs the individual devices to initialize. Further, it configures the starting values of a certain scenario at the individual devices.
3. If a condition is specified, the simulation runs until this condition is fulfilled. For instance, it is demanded to run a simulation until 20 products have been produced successfully.
4. If a scenario collection consists of various scenarios, the simulation runs until the simulation of each scenario collected has been realized.

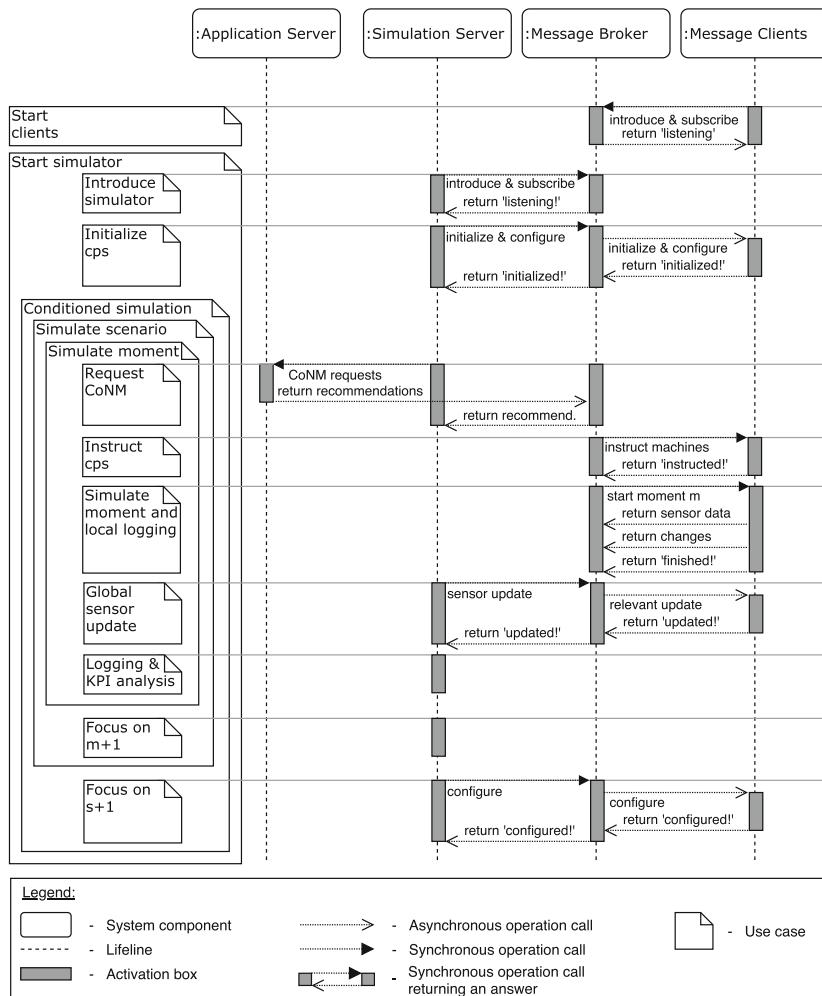


Figure 7.9 The System Interactions Arranged in Time Sequence (as UML Sequence Diagram Variant)

5. If a scenario consists of various moments, the simulation runs until the simulation of each moment has been realized. The simulation of a moment considers the procedure steps 6–10.
6. If desired, the CoNM is requested for recommendations and the CoNM results are considered at the simulation system. The implementation of the LEGO simulator has considered the CoNM integration as follows: According to the learning task specified by the NMDL, any kind of virtual or physical sensor will be connected to the CoNM messaging channel. The simulation server collects transmitted parameters and prepares the CoNM request.
7. Depending on the current scenario, devices are instructed to realize a certain process or a network of processes individually.
8. The current moment is started by the simulation server. Each device simulates its moment autonomously. Here, it processes environmental simulation data and system-specific data, carries out actuator instructions and collects relevant sensor data. If defined as global sensor data, the system publishes environmental changes to the joint simulation system. Finally, a message indicates the finalization of the device-specific simulation.
9. When any moment is simulated, environmental changes of other devices are perceived and updated by the individual device.
10. The simulation activity of the individual devices is logged. The performance is published at the very end of the simulation runs.
11. The next moment of a scenario is focused if the scenario consists of further moments to be simulated.
12. The next scenario of a collection of scenarios is focused if the collection consists of further scenarios to be simulated. If the initial condition of a simulation target has not been satisfied by this scenario, the scenario collection is appended by a further scenario. Before the next scenario starts, the devices will be configured for the next scenario.

e) Performance Measurement: In order to compare or rather benchmark simulation runs realizing different instruction strategies, a framework for the analysis of the CPPS performance has been set up. This has been developed on the basis of the design-oriented approach for the search of a global optima of Grum, Bender, and Alfa (2017) and the performance evaluation framework of Grum et al. (2018). So, the following system-specific KPIs were conceptualized.

- **Production Outputs:** The number of production outcomes o_i^t refers to the collection of workpieces that have been realized by the activity of CPS i at the time

step t . Since the machines might realize an outcome even after being disturbed by an alarm situation causing failures, for instance, because a maintenance worker has disturbed the regular production routine, we distinguish between the correct production outcomes realized at a certain time step or moment t called $o_i^{t,+}$ and the incorrect production outcomes $o_i^{t,-}$.

- **Initialization Attempts:** The number of initialization or rebooting attempts r_i^t refers to the collection of reboots that have been required for CPS i in an alarm situation at the time step t . Since the machines might initialize or reboot although this kind of activation was not required, for instance, because a cleaning operation falsely activates a sensor and causes an alarm situation, we distinguish between the correct initialization attempts realized at a certain time step or moment t called $r_i^{t,+}$ and the incorrect initialization attempts $r_i^{t,-}$.
- **Alarm Signals:** The number of alarm signals a_i^t refers to the collection of alarms that have been caused by the activity of CPS i at the time step t . Since the machines might cause an alarm signal despite it not being an alarm situation, for instance, because a cleaning operation falsely activates a sensor and disturbs the regular production routine, we distinguish between the correct alarm signals caused at a certain time step or moment t called $a_i^{t,+}$ and the incorrect alarm signals $a_i^{t,-}$.
- **Productive Moments:** The active time or productive moment p_i^t refers to the period that begins at the time step t at which a job is realized at a CPS i and ends at the next time step or moment $t + 1$. Since the machines might be instructed to realize a production task without having the components required, for instance, only the milling motors are started without having the object to be milled, we distinguish between the correct productive moments occurring at a certain time step or moment t called $p_i^{t,+}$ and the incorrect productive moments $p_i^{t,-}$.
- **Idle Moments:** The waiting time or idle moment w_i^t refers to the period that begins at the time step t at which a pause is realized at a CPS i and ends at the next time step or moment $t + 1$. Since the machines might be instructed to wait despite not being able to realize a production task correctly, for instance, a milling machine pauses even though an object to be milled waits for the milling process, we distinguish between the correct idle moments occurring at a certain time step or moment t called $w_i^{t,+}$ and the incorrect idle moments $w_i^{t,-}$.
- **Production Costs:** The processing or production cost $c_{p,i}^t$ refers to the cost that the processing incurs at CPS i during a productive moment p_i^t . As one assumes these to develop over time because of market mechanisms, the machine-specific production costs per moment $m_{p,i}$ are time-dependent and dependent on more versatile task types. These are initially assumed to be constant. Since the machines might be instructed to realize a production task without having the

components required, for instance, the milling program of a machine is started without having the object to be milled available, we distinguish between the correct costs occurring at a certain time step or moment t called $c_{p,i}^{t,+} = p_i^{t,+} \cdot m_{p,i}$ and the incorrect costs $c_{p,i}^{t,-} = p_i^{t,-} \cdot m_{p,i}$.

- **Idle Costs:** The waiting cost or idle cost $c_{w,i}^t$ refers to the cost that the processing incurs at CPS i during an idle moment w_i^t . As one assumes these to develop over time because of market mechanisms, the machine-specific idle costs per moment $m_{w,i}$ are time-dependent. These are initially assumed to be constant. Since the machines might be instructed to wait although a production task could be realized, we distinguish between the correct costs occurring at a certain time step or moment t called $c_{w,i}^{t,+} = w_i^{t,+} \cdot m_{w,i}$ and the incorrect costs $c_{w,i}^{t,-} = w_i^{t,-} \cdot m_{w,i}$.

Since the benchmark intends to focus on the KPIs across systems addressing the search for a global optima, we are interested in the global KPIs summarizing all the moments t and all the scenarios s required to fulfill the condition that has been specified initially. An example is the successful production of 20 outcomes. The assessment of efficient production task processing is based on the total KPIs $K*$, resulting in the use of Eq. (7.1):

$$K* = \sum_{s=1}^S \sum_{t=1}^{M_s} \sum_{i=1}^N k_{i,t,s}* \quad (7.1)$$

where N refers to the number of production systems, M_s is the number of moments of a scenario s , S is the number of scenarios being part of the simulation run and $k*$ refers to any kind of system-specific KPI. So, we have the correct and incorrect versions of the total idle costs C_w , the total production costs C_p , the total idle moments W , the total productive moments P , the total alarm signals A , the total initialization attempts I and the total production outputs O .

Data Collection: The data collection considers the environment provided by the Lego simulator. Here, a conditioned simulation has been realized, which lasted as long as the condition $O_4^+ = 100$ has not been fulfilled. Since production outputs $o_i^{t,+} \forall i = 1, 2, 3$ were interpreted to be interim products, the successful output was identified at *EV3_04* if the previous production steps have been realized without disturbance.

Parameters assumed per simulation run were discussed with eight production managers of SMEs in a workshop session, because of which the creation of a realistic setting has been supported. Machine-specific production costs per moment were set

to $m_{p,i} = 450.00\text{€}$ $\forall i$, and the machine-specific idle costs per moment were defined to be $m_{w,i} = 150.00\text{€}$ $\forall i$. Costs of differing value were determined to be interesting but only for later simulation runs. Further, rates were specified as follows: In eight out of ten scenarios, a workpiece will be produced so that the production rate refers to 80%. In one out of ten scenarios, a maintenance will be realized so that the maintenance rate refers to 10%. In two out of ten scenarios, a cleaning will be realized so that the cleaning rate refers to 20%. So, focusing on work week, for example, the production will be productive in four out of five days. The maintenance was realized every two weeks, and the cleaning of the joint production hall is realized once a week, which is on Friday, so that the production system is ready for the next week before the weekend. Hence, the scenarios occurred as Tab. 7.4 shows.

Table 7.4 Lego Scenario Probabilities

Scenario Denomination		Probabilities				
ID	Title	Cleaning	Workpiece	Maintenance	$P_{ID}(W, M)$	$P_{ID}(C, W, M)$
1	Idling the production	0.8	0.2	0.9	0.18	0.144
2	Maintaining the robot	0.8	0.2	0.1	0.02	0.016
3	Producing the workpiece	0.8	0.8	0.9	0.72	0.576
4	Production and maintenance in conflict	0.8	0.8	0.1	0.08	0.064
5	Cleaning causes maintenance program	0.2	0.2	0.9	-	0.036
6	Undisturbed cleaning and maintenance	0.2	0.2	0.1	-	0.004
7	False alarm causes production stop	0.2	0.8	0.9	-	0.144
8	Undisturbed cleaning and alarm situation	0.2	0.8	0.1	-	0.016
		Total Probability			1.000	1.000

In the table, the individual scenarios have been referred to in accordance with the bivalent Truth Table shown in Tab. 7.3, such that the scenarios can be identified by the same IDs or titles. Probabilities of the random variables of the occurrence of a cleaning workpiece being produced or maintenance can be seen at the right of the

corresponding scenario. The manifesting probability of a certain scenario P_{ID} has been positioned at the very right of the table. Please note here the two different kinds of probabilities. This will address different time points of a data collection which will be explained in the following by the CoNM proceeding phases.

For the practical application of the CoNM at the Lego simulator realized, the twelve phases of the CoNM proceeding have guided the project realization (cf. Fig. 4.9). The following will present the outcomes phase-wise.

While the project initiation has been omitted because of the nature of the simulation, the target agreement has determined the optimization of the production process spanning the four kinds of CPS called *EV3_01 - EV3_03*. This refers to the process presented in Fig. 6.27 and must include the subsequent production tasks of *EV3_04*. The performance of the ANNs constructed will be measured by the KPIs specified in Eq. 7.1. Detailed outcomes of the third phase about the process modeling and the fourth phase about the knowledge modeling can be found in section 6.5 (cf. Fig. 6.28 and Fig. 6.29).

During phase 5, the data has been collected from the simulation system. So far, this has been based on the standard scenarios having the IDs 1–4 (see $P_{ID}(W, M)$ at Tab. 7.4). Further, the live system has been connected via workflows to the knowledge objects of the NMDL's *Set View* (see Fig. 6.31). Grace to this, the physical systems directly can message data in the CoNM system. More precisely, the ANN system that controls the robot and has been examined in section 6.5 has been connected with the input data streams of *sensor 12* as *input 1* (IDs according to Tab. 7.2) and *sensor 17* as *input 2*. The alarm system that unleashes the production system has been examined to be optimized in section 6.6. It has been connected with the input data streams of *sensor 15* as *input 1* (IDs according to Tab. 7.2) and *sensor 18* as *input 2*.

During the sixth phase, the ANNs have been trained on the base of data collected (cf. fixed learning task, Def. 21). This has been generated by the standard environment of the simulation system. Details about the final ANN, such as the robot network compartments, have been presented in Fig. 6.30. Its behavior can be seen at Fig. 6.31 (b). Details about the alarm system network can be seen in Fig. 6.35 (a) and Fig. 6.36 (a). The corresponding behaviors resemble the one presented in Fig. 6.17.

During the seventh phase, the ANNs have been used as a decision system directly controlling the machines of the Lego simulator. This can be interpreted as a first simulation run, whose performance can be compared with the standard simulation environment. Having realized the two kinds of simulation runs, troubles were solved, such as machine breakdown of the Lego arm because of a broken gearwheel and cable renewing because of broken cables. The messaging and the integration of

CoNM recommendations has been proven to be stable during all the simulation runs.

During the run-time, continuously, the CoNM views have been inspected when activated. In detail, the inspections have been explained in section 6.4.1 (static knowledge identification) and section 6.4.2 (dynamic knowledge identification). So, the concrete operation of the production system was made transparent. On behalf of the simulations because of the CoNM's NPS, the next moment of the production system was predicted. On behalf of the CoNM recommendation, the optimal next production moment for the systems of *EV3_01* and *EV3_02* was determined. This has been used to instruct the machines. Hence, on the one hand, we can determine the production performance of the standard simulation environment. On the other hand, we can determine the performance of the production system being controlled by the NPS-based simulation. Since the NPS-based simulation is used to collect the data from the operation of the standard environment, it shows that both kinds of simulation runs show the very same performance.

In order to confirm the exception stated, 100 products successfully have been produced by the two simulation environments mentioned. In regard with the metaphor of one workweek, the first simulation runs have simulated about 6.75 months at the standard environment and about 6.65 month at the NPS-based environment until the production target of 100 successful production outcomes has been reached per strategy. In total, this simulation has lasted for about 3.2 hours running time.

By comparing these two kinds of simulation environments, it was determined that the production system has manifested the same observable behavior for both kinds of environments. So, independent from the choice of the system instructing the machines, the simulation has manifested as it was described in the first four scenarios of the *scenario design* presented above. From a visual viewpoint, it therefore does not matter if either the standard simulation environment or the NPS-based environment instructs the machines. This accounts for all the standard scenarios without exception.

In regard with the KPIs specified, the visual impression was confirmed: it does not matter if either the standard simulation environment or the NPS-based environment has instructed the machines, because the simulation runs have lead to the same performance level. While the following issues the most relevant analyses and interpretations, the detailed KPIs can be found in Tab. D.2 and diagrams are shown in appendix D.3–D.6.

Facing the scenario distributions, one can see that the scenarios occurred within the standard environment (Fig. D.2) and the NPS-based environment (Fig. D.3) resemble the ideal distribution of the scenarios (Fig. D.1), which is in accordance

with the probabilities of Tab .7.4. Hence, the following interpretations are representative.

Addressing the time of the production of 100 products, both kinds of simulation environments have required about the same number of production moments and scenarios (cf. Fig. D.4). Having a deeper look at the kinds of moments, it can be concluded that the number of correct production moments and correct idle moments is similar as well (cf. Fig. D.5). Please note here that the number of incorrect production moments and incorrect idle moments are 0, because the disturbing factor of the cleaning process does not occur in the standard scenarios. Since the CoNM has been requested for instructing the machines in the NPS-based environment, this additional processing time has led to a longer simulation time of the NPS-based environment (cf. Fig. D.6).

Addressing the quality of the production of 100 products, only products have been produced without any quality defects (cf. Fig. D.7). Incorrect products have not been produced because no disturbance factor has led to an erroneous incident. Hence, the number of alarm calls (Fig. D.8) and the number of caused re-initialization attempts are comparable in terms of the total numbers and correctness (Fig. D.9).

Addressing the cost of the production of 100 products, total costs of the NPS-based environment are slightly lower than the costs of the standard environment (Fig. D.10). This can be traced back to the higher number of unproductive scenarios (scenario 1 and scenario 4) in the standard environment. By having a deeper look at the idling costs at Fig. D.11, this impression is confirmed.

Let us assume an environmental change now. An additional disturbance factor has come up, so the cleaning scenarios are present in addition (see $P_{ID}(C, W, M)$ at Tab. 7.4). This refers to a modification of the initial fixed learning task (Def. 21). Since now the cleaning machine stops the production system when a workpiece is to be produced, in not any standard scenario will an output be produced successfully. In order to verify the working of the production system, the maintenance worker therefore has overruled the alarm system, so that a workpiece can be produced despite the presence of an alarm situation. In this changed setting, during run-time, data has been collected for the scenario with the IDs 5–8, which has served for the neuronal optimization and corresponding modifications of the tenth phase. Details of both the NPO-based models have been described in section 6.6. Since both kinds of NPO approaches have led to the same process modification, a second simulation run will verify the improvement caused by this.

In order to confirm the exception stated, the conditioned simulation has been extended because of the higher number of scenarios. So, the second simulation runs have lasted as long as the condition $O_4^+ = 200$ has not been fulfilled for any of the three simulation environments, that is, the standard environment, the NPS-

based environment and the NPO-based environment. Regarding the metaphor of one workweek, the second simulation runs have simulated about 19 months within the standard environment, about 18 months at the NPS-based environment and about 14 months at the NPO-based environment until the production target of 200 successful production outcomes had been reached. In total, this simulation has lasted about 13.4 hours running time.

During the second simulation runs, it was determined that the production system has manifested the same observable behavior when it is instructed by the standard simulation environment and the NPS-based simulation environment. So, from a visual viewpoint, it does not matter if either the standard simulation environment or the NPS-based environment has instructed the machines. Since these two kinds of simulation environments have not adapted to the environmental changes, they lead to the same bad behavior shown by the cleaning disturbed scenarios, which has been described in the *scenario design* presented above. As opposed to them, the NPO-based environment successfully has overcome the disturbing influence of the cleaning machine to improve its behavior. Since it has adapted to the environmental changes, it instructs machines in the cleaning scenarios as if these were standard scenarios. Hence, production outputs are produced here, although the standard environment and the NPS-based environment run in an alarm situation.

Regarding the KPIs specified, the visual impression was confirmed because it does not matter if either the standard simulation environment or the NPS-based environment has instructed the machines, as both the environments operate on the same performance level. However, the best candidate to instruct machines is the NPO-based environment. While the following issues the most relevant analyses and interpretations, the detailed KPIs can be found in Tab. D.3 and diagrams of the second simulation runs are presented in appendix D.8–D.11.

Facing the scenario distributions, one can see that the scenario distributions of the standard environment based run (Fig. D.12(b)), of the NPS-based environment run (Fig. D.12(c)) and the NPO-based environment run (Fig. D.12(d)) resemble the ideal scenario distribution (Fig. D.12(a)), which is in accordance with the probabilities shown in Tab. 7.4. Slight deviations can be found at a slightly smaller occurrence frequency of the productive scenario 3 and a slightly bigger occurrence frequency of the unproductive scenario 7 of the standard environment. This effect would diminish because of the law of large numbers.

Addressing the time of production of 200 products, the NPO-based environment clearly required less moments than the NPS-based and the standard environment (cf. Fig. D.13). Here, one can recognize the following. While the latter kind of environment required slightly more unproductive scenarios as it was mentioned before, the standard environment slightly required more moments than the NPS-based environ-

ment. This effect would diminish after many simulation runs. A deeper look at the moments (Fig. D.14), confirms this impression. The standard and the NPS-based environment clearly required more productive and more unproductive moments than the NPO-based environment. This includes the differentiation between correct and incorrect moments. Here, the NPO successfully has avoided incorrect productive and idling moments, because the disturbing cleaning machine effect has been canceled out. Regarding the time the simulation runs have lasted (cf. Fig. D.15(a)), one can recognize the extra time that was required by the CoNM recommendation engine. Normalized over all scenarios which isolates the effect of different numbers of moments, both the CoNM requests required the same amount of time (cf. Fig. D.15(b)).

Addressing the quality of the production of 200 products, one can see the benefit of the NPO-based environment, whose instructions have led to the avoidance of quality defects in products caused by erroneous alarm situations. Again, the NPS-based environment and the standard environment resemble the production of output fails (cf. Fig. D.16). The very same kind of impression can be found in the occurrence of alarm signals (cf. Fig. D.17) and initialization moments because of alarm signals (cf. Fig. D.18).

Addressing the cost of production of 200 products, one can see the least costs involved in the NPO-based instruction strategy (cf. Fig. D.19). The slight deviation in the reduced costs of the NPS-based environment from the standard environment can be traced back to the smaller occurrence frequency of the productive scenario 3 and the bigger occurrence frequency of the unproductive scenario 7 of the standard environment mentioned before. By having a deeper look at the cost characterization in Fig. D.20, one can recognize the same kind of incorrect production and idle costs as well as correct production and idle costs within the NPS-based and the standard environment. The NPO-based environment successfully has avoided costs because of incorrect production and idle process instructions and does cause correct production and idle costs only.

Summarizing the insights of all the simulation runs, one can confirm the same kind of visual behavior as well as KPI-based performance of the NPS-based environment instructed machines as the standard environment. Further, the NPO-based instructions outperform the previous two kinds of environments in terms of the visual performance as well as the KPI-based performance.

Potential: Coming to the Lego simulator implemented and the CoNM integration shown, the CoNM potential becomes transparent by three criteria as follows:

a) Usefulness: The usefulness can be clarified by the following aspects:

- (1) Since any industrial production task can be simulated with the LEGO simulator, and the simulator has been connected with the CoNM main artifact, the CoNM main artifact has been evaluated with regard to the practical applicability of the CoNM.
- (2) The CoNM has demonstrated to realize a simulation, which leads to a correct production system behavior without exceptions.
- (3) Since the different production strategies have been evaluated on an equal footing, the Lego simulator has provided a common base for the performance comparison. So, the successful production of workpieces has been evaluated by facing the CoNM.
- (4) Beyond the benchmark that is based on the Lego simulator presented, the CoNM's NPO has been identified as the superior strategy to instruct machines. It can therefore stand as a new foundation for management and controlling instances of CPPS.

b) Generality: Having considered an arbitrary simulation system, the evaluation claims for generality with regard to the first application of the CoNM main artifact in reality. This CoNM application therefore can stand as the first reference point.

c) Novelty: With the aid of the Lego simulator, the ANN Process Domain has been made transparent for the first time in a real setting, and the CoNM successfully has controlled and improved a real process network. Regarding the application results of the CoNM constructed, it became clear that the research gap has been addressed successfully on a valid level.

Back References: The *evaluation question*, that is, (“To what extent can the CoNM be applied to a real system?”), can be answered in terms of the problems an application has not been able to manage. The simulation has shown that the CoNM has been applied to a real production system, live simulation settings and third party devices successfully. Since the time to derive CoNM recommendations may slow the production system, please compare here the production moments simulated with the simulation time, following CoNM versions will face the speed-optimized calculation of CoNM recommendations. So far, all the calculations required have been realized by a single laptop only, which further has served as a message broker server and simulation server.

The practical applicability of the CoNM to real systems has been captured as follows. First, the CoNM has been able to model real systems with the aid of neuronal processes. Second, the NPS has been able to stand as a simulation system for a real system. The physical devices have shown the same behavior as if they were controlled by the standard simulation environment. Third, the NPO has been identified to be superior in a benchmark with the standard simulation environment. It has needed less moments to produce the same amount of production outcomes successfully. Further, it has lowered the costs. The *evaluation purpose* so has been reflected.

The *evaluation objective* of a practical applicability has been reflected since problems of an integration with a real production system have been issued. Since the CoNM-based simulation has lead to the same behavior than the standard simulation environment, the purpose of a simulation system has been satisfied by the CoNM's NPS. The difference is that the NPS-based simulation system has been constructed by data-based training routines autonomously, while the standard simulation environment has been implemented by high implementation efforts. Furthermore, the neuronal simulation has been optimized by data-based training routines autonomously. Please note here that an autonomous optimization cannot be realized by standard simulation environments. In these two points, the CoNM-based simulator is superior to a standard simulator environment.

In accordance with the requirements provided in section 3.1.3, the CoNM enables the integration with real systems. Hence, the CoNM has proven to apply the concept of neuronal modeling to real systems.

7.3 Attempts For Coherentism & CoNM-Contribution

As the individual conclusions of each sub-artifact have been presented (section 7.1.2) and the empirical evaluation confirms (section 7.1.3 and section 7.2), the following intends to integrate separate evaluations. It intends to regard them within the context of neuronal modeling and the research gap addressed. For this, the following structures by the criteria of the concept matrix shown in Tab. 2.3, with which the research gap of a neuronal modeling was identified within the context of a SLR (section 2.7). Each criteria has been highlighted by a bold type. Hence, the guiding question is to what extent can the joint consideration of the sub-artifacts as modeling concept of a CoNM coherently address the research gap identified.

Modeling Language: Based on the generic meta-model designed in section 4.1 and having specified a technical foundation by the CoNM at section 4.6, a modeling

language manifested as the CoNM modeling language. This has been referred to as NMDL (cf. section 4.6.4) and interrelates the *Process Domain* as well as the *ANN Domain*. Hence, both modeling language specific sub-criteria shown in Tab. 2.3 can be satisfied. As opposed to contemporary modeling languages, the CoNM artifacts jointly allow the following:

- ⇒ The CoNM artifacts allow the model specification of any level of process networks, processes, tasks, activities or activations as ANN-based system. Since it functions as a learning system, it supports the autonomous model construction; the CoNM identifies relations at the level considered, which is on behalf of data faced during the course of training.
- ⇒ The CoNM artifacts provide a technical foundation so that the process systems are able to adapt to environmental changes. This is based on the consideration of training data.
- ⇒ The CoNM artifacts provide a technical foundation so that the process systems are able to react on environmental changes. So, they are able to instruct any kind of organizational entity via workflows.
- ⇒ The CoNM artifacts provide a technical foundation so that data input coming from external simulators or third party systems or being relevant for these kinds of systems is buffered by the modeling language.

Knowledge Representation: Since the SEKO concepts have identified various forms of knowledge objects as well as neuronal conversions (cf. experiments at section 6.4), knowledge exemplary has been recognized in both, human as well as ANN-based knowledge carriers. By realizing the symbiotic KM activities (cf. section 4.5), the systematic identification of an organization-wide knowledge base as well as the harmonized knowledge representation is supported. Hence, both knowledge representation specific sub-criteria of Tab. 2.3 can be satisfied. As opposed to contemporary approaches to represent knowledge, the CoNM artifacts jointly allow the following:

- ⇒ The CoNM artifacts provide a technical foundation so that the biologically plausible approximation of human as well as machine-based knowledge carriers is enabled. This is integrated with a joint processual understanding of actual entities, so that any kind of object can be approximated by CoNM mechanisms.
- ⇒ The CoNM artifacts comprehend NMDL-based models, which therefore function as communication platforms; the NMDL visualizes internal and external

input data for ANN systems and further visualizes the knowledge generation, use and transfer of ANN systems.

- ⇒ The CoNM artifacts provide a technical foundation so that knowledge directly can be extracted from ANN-based systems. Structural as well as dynamic forms of knowledge have been identified.
- ⇒ The CoNM artifacts realize a constructive knowledge understanding so that knowledge can be examined on various hierarchical levels. So, domains that recognize knowledge at higher cognitive levels, such as the traditional KM, and domains that recognize knowledge at lower cognitive levels, such as the neuroscience, can be harmonized.
- ⇒ The CoNM artifacts objectify knowledge so that knowledge objects can be reused for other ANN systems without limiting the operational working of ANN-based systems.

Simulation: By following the CoNM proceeding (cf. section 4.4) and building on the simulation concept of the CoNM (cf. section 4.6), relevant CoNM concepts have been implemented. With the aid of the modeling tool called *Modelangelo* and the library called *PyBrain* (cf. the implementations at section 5), a NEA server has been set up (cf. section 4.3) that carries out process simulations, ANN simulations as well as neuronal process simulations. Hence, both simulation specific sub-criteria shown in Tab. 2.3 can be satisfied. As opposed to contemporary simulation systems, the CoNM artifacts jointly allow the following:

- ⇒ The CoNM artifacts allow an approximation of ANN-based system function at an arbitrary detailed level so that the simulation accuracy can be increased to an arbitrary high accuracy. This is comparable with the arbitrary detailed function approximation of Hornik, Stinchcombe, and White (1989).
- ⇒ The CoNM artifacts allow the consideration of the joint simulation system to be an ANN-based system. So, the simulation system can be constructed autonomously on behalf of data during the course of training.
- ⇒ The CoNM artifacts allow the approximation of selected systems on behalf of ANN-based mechanisms so that simulated systems as well as the joint simulation system follow (more or less) biologically plausible models.
- ⇒ The CoNM artifacts provide a technical foundation so that any form of input data, output data and data addressing the inner working of ANN are visualized by the NMDL. Hence, these kinds of data can be accessed via the objectified forms of objects by their modeling items that are combined with novel forms of datasets for instance. This enables the on-building scenario specification as well as the subsequent training and testing specifications.

Optimization: By following the CoNM proceeding (cf. section 4.4) and building on the optimization concept of the CoNM (cf. section 4.6), relevant CoNM concepts have been implemented. With the aid of the modeling tool called *Modelangelo* and the library called *PyBrain* (cf. the implementations at section 5), an NEA server has been set up (cf. section 4.3) that carries out process optimizations, ANN trainings as well as neuronal process optimizations. Hence, both optimization specific sub-criteria shown in Tab. 2.3 can be satisfied. As opposed to the contemporary optimization procedures for process improvements, the CoNM artifacts jointly allow the following:

⇒ The CoNM artifacts allow the consideration of any level of process networks, processes, tasks, activities or activations as ANN-based system. Here, it functions as an optimizing system which identifies modifications of the level considered autonomously on behalf of data during the course of training.

Learning: In order to realize NPM, NPO and NPS, contemporary learning principles, or rather training procedures, have been transferred to the *ANN Process Domain*. Further, new learning principles have been designed (cf. section 4.8). With the aid of the modeling tool called *Modelangelo* and the library called *PyBrain* (cf. the implementations at section 5), a NEA server has been set up (cf. section 4.3) that provides these kinds of new learning procedures. Hence, both learning specific sub-criteria shown in Tab. 2.3 can be satisfied. As opposed to contemporary training procedures for ANN systems, the CoNM artifacts jointly allow the following:

⇒ The CoNM artifacts allow the ANN-based learning in the *ANN Process Domain*.
⇒ The CoNM artifacts allow the learning over crystallized structures such as pre-trained process networks, processes, tasks, activities, activations.
⇒ The CoNM artifacts allow the optimization of any kind of model (process models, business process models, simulation models, ANN models) with the aid of novel forms of ANN-based learning mechanisms.

Interim Conclusion: On comparison with the research gap shown in Tab. 2.3, one can identify that the CoNM as modeling concept can satisfy the research gap because of the fulfillment of the analysis criteria considered. Further, it is different from the state-of-the-art and provides the collection of research contributions presented.



Concluding Remark

8

The concluding remarks will provide a summary of the research carried out. The results and insights have been appraised critically so that it becomes clear as to how far the research questions have been answered and the limitations have been identified, or which kind of research work needs to be addressed next. According to Peffers et al. (2007), this refers to the sixth step of a design-oriented research proceeding and the following sections support the communication process.

8.1 Summary

The CoNM is a methodological approach which enables model systems to adapt quickly, efficiently and autonomously to environmental changes. This does not only count for natural systems such as human knowledge-bearers, tangible and non-tangible objects such as resources or prestige, and their interacting in process networks. This further includes artificial systems such as ANN approximating any kind of modeling objects. Particularly for the target-oriented construction, management and evaluation of those systems, problem-specific and adequate models, proceedings and methods must be provided.

Since the variety of methods for process and knowledge management, as well as the simulation and ANN training approaches is huge, and an adequate addressing of the neuronal process modeling has not been realized yet, this contribution interrelates various domains and creates an integrated *ANN Process Domain*. Fig. 8.1 visualizes the fusion of different theories and domains.

Supplementary Information The online version contains supplementary material, such as Appendix A, B, C and D whose references are indicated in the text, available at https://doi.org/10.1007/978-3-658-35999-7_8.

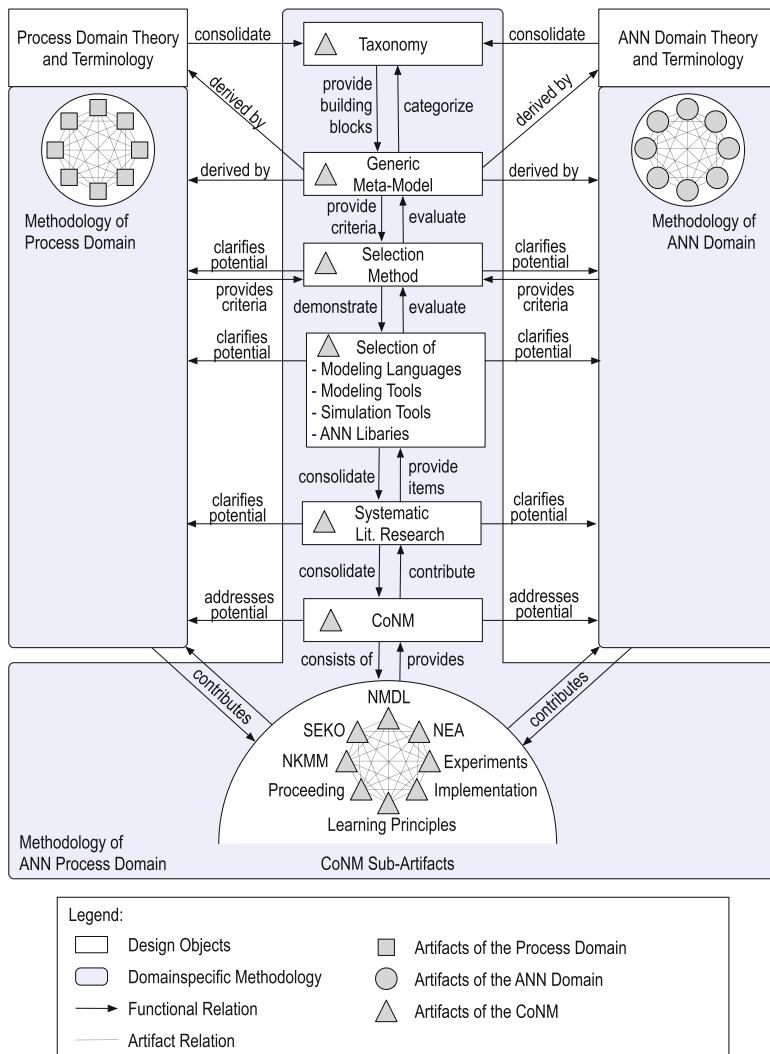


Figure 8.1 CoNM Artifacts Realized by this Research

For the differentiation of domains and the precise specification of CoNM, a modeling concept has been worked out to function as *reference framework*, which provides relevant elements and their associations in the context of ANN-based process modeling. So, an overall understanding of ANN-based neuronal process systems and its taxonomy has been established. Ideally, process and business process elements and ANN mechanisms are considered in harmony. This contribution presents an explanatory model for neuronal process modeling, which makes this equilibrium healthy. It manifests as the so called generic meta-model.

In order to identify the missing elements and associations, as well as to evaluate contemporary research artifacts of relevant domains, a *selection and evaluation method* was created, using which the state-of-the-art was analyzed. The *reference framework* developed has been complemented by eight novel artifacts that specifically address the research gaps identified using the *selection and evaluation method*. Since the framework is more powerful if it includes further methods, the triangular shapes of new artifacts have been surrounded by a semi-opened circle in Fig. 8.1. Therefore, these represent the least required number of relevant artifacts of a CoNM. The artifacts have been edited in a way that they can be combined using a mixed-methods-based design during the demonstration phase. Each of the following artifacts has been created by an individual research cycle. Namely, the artifacts refer to the following:

NMDL stands for neuronal modeling and description language. *SEKO* stands for the concepts to systematically extract knowledge objects from ANN operation. The *Learning Principles* enable the neuronal simulation and neuronal optimization by establishing an adequate understanding of ANN in the corresponding contexts. Further, novel training approaches have been set up here. The *Proceeding* set up clarifies the steps to realize at the CoNM use. The *NKMM* structures the symbiotic knowledge management between human and ANN-based knowledge carriers. The *NEA* refers to the neuronal enterprise architecture that is required for CoNM implementation and its software prototypes. The experiments prove the functioning of artifacts created and demonstrate their use.

The first demonstration clarified how to use the selection method designed in the context of an SLR. In the second run, the very same method was used to evaluate the joint CoNM. It was identified that the CoNM has progressed the current state-of-the-art and has reduced the variance in its evaluation. The second demonstration clarified by theoretic models as to how to set up contemporary ANN architectures with the NMDL. With the aid of sic experiments, artifacts have been functionally proven as follows.

The *first experiment* addressed the specification of ANN as well as the NMDL substantiation by ANN mechanisms. This has lead to a rise of transparency when dealing with ANN.

Two further experiments demonstrated how to carry out a neuronal process modeling, because of which tacit knowledge can be identified on a neuronal level by CoNM mechanisms automatically. Using this, neuronal conversions can be recognized, and the knowledge construction as well as the behavioral perspective become identifiable. This has further increased the transparency because it lightens the inner knowledge generation by ANN.

One experiment demonstrated how to set up neuronal process simulation models. The realization of the simulation has lead to plausible results. Further, this has increased the transparency in the inner working of ANN, because it enables the recursive top-down or bottom-up modeling by human- as well as machine-based model constructors.

Two further experiments have addressed regarding how to carry out a neuronal process optimization. Since there are two variants per experiment, in total, four optimization runs have lead to plausible results. This has increased the transparency as well since the inner working of ANN during the course of learning has been directed via architectural levels that are comparable to more or less complex cognitive structures. Since process parts have been identified to be left out if they are not relevant for the successful process realization, this has lead to a saving of organizational resources.

The CoNM constructed by this contribution refers to models for processes, neurons, simulations and optimizations as well as their technical realization by ANN mechanisms. It enables the system supervisors to efficiently analyze, evaluate, design, simulate, optimize and manage any form of processes and knowledge bearers. The *practical applicability* of the CoNM-enabled activities has been proven in a real setting, which referred to a LEGO-based production system.

8.2 Critical Appraisal

The process of the CoNM modeling concept creation presented in Fig. 8.1 will be appraised critically here. This includes considerations about the methodological proceeding, the overall results as well as single results. Faced with a mixed-method research design, one must note that the overall result is based on a variety of insights of sub-artifacts.

8.2.1 Research Questions

Having identified a research gap during the modeling of processes by using neuronal techniques, structured insights have been identified by the collection of four research questions. Corresponding to the objectives presented per research question in section 1.2, the following identifies answers for each research questions proposed.

First Research Question: The first research question, that is, (*How can the modeling capability of knowledge modeling description languages be used to raise the transparency of NN?*), can be answered as follows: The NMDL designed increases the transparency as the inner working of NN is enlightened. This includes its structural design, as well as its dynamic behavior within the temporal course. Further, the identification of the kind of system is issued so that it becomes clear which approximated resources or process participants take part in the realization of a process. Since the NMDL allows the specification of architectural, location-based architectural levels, and the CoNM learning principles support these, the building of hierarchical systems and more or less complex cognitive structures is enabled. Hence, the transparency is increased by this as well. Considering ANN as knowledge carriers, the system-specific generation of knowledge can be issued. For this, first a meta-model has been designed, from which the modeling syntax was derived. The final modeling shape specification has lead to the *NMDL* manifestation.

Since the NMDL modeling items have been technically substantiated by ANN mechanisms, process models can be mapped to neuronal processes, and the NEA prepares the CoNM tool implementation.

Faced by the term “use” of the sub-research question proposed, the *CoNM Proceeding* was set up; it specifies how to use the NMDL. As the NMDL intends to deal with both, human-based and neuronal knowledge, the symbiotic knowledge management model called *NKMM* was set up. It explains the interplay of various experts, activities to be carried out and the management levels that need to be addressed by pedagogical stages.

Second Research Question: The second research question, that is, (*How can the learning capability of NN be used to identify tacit knowledge?*), can be answered as follows: The learning capability of ANN makes ANN structures fit the reality regarding the behavior of objects selected. As the inner working of ANN is made transparent by the CoNM, one can identify knowledge transfers among neurons at the very atomic level, because of which the tacit knowledge becomes identifiable. Here, two approaches have been designed in order to identify knowledge, which cannot be seen in reality but is used within the ANN. Denominated as “System-

atic Exhaustion of Knowledge Objects” (SEKO), the first focuses on the structural knowledge of ANN and the second focuses on the dynamic knowledge of ANN.

Through CoNM implementation, the SEKO approaches can be specified and initiated by modeling tools such as the *Modelangelo*. With the aid of ANN libraries such as *PyBrain*, the CoNM tool carries out NPM tasks. In accordance with the NEA designed, the interplay of various systems is harmonized for this. Hence, SEKO results can be projected on NMDL models, because of which the knowledge transfers and tacit knowledge of ANN become transparent, and the NMDL serves as communication platform for human- and machine-based model constructors.

Third Research Question: The third research question, that is, (*How can the modeling of neuronal processes be used for process simulations?*), can be answered as follows: As modeling items characterize a simulation scenario with the aid of the NMDL designed, and further, the NMDL is technically substantiated by ANN mechanisms, process simulations can be carried out by activating neuronal processes.

Through CoNM implementation, the NPS runs can be specified similar to NPM runs: They are initiated by modeling tools and implemented by ANN libraries. The CoNM tool integrates them and carries out NPS tasks. In accordance with the NEA designed, the different kinds of systems are connected with the CoNM. So, NPS results can be projected on NMDL models, because of which the ANN behavior as well as the knowledge construction become transparent and the NMDL serves as communication platform for human- and machine-based simulation model constructors.

Fourth Research Question: The fourth research question, that is, (*How can the learning capability of NN systems be used for process optimizations?*), can be answered as follows: As the structure and tempo-spatial behavior of ANN systems is integrated with modeling objects of the NMDL, ML procedures can directly affect NMDL-based models; when an ANN system is affected because of training-based modifications, the corresponding modeling items such as process networks, processes, tasks, activities, activations, etc. are affected as well. Experimentation results showed that these changes can lead to process optimizations, because of which an extension of experiments seems necessary in order to generalize correctly. Here, an NPO training can either focus on higher cognitive systems or rather more complex systems, for which an *NPO Backpropagation* training was designed, or an NPO training can focus on the modification of any systems, for which standard *Backpropagation* procedures were designed in NPO contexts.

Through CoNM implementation, the NPO runs can be specified and initiated by the *Modelangelo* and realized with *PyBrain*, that refer to two example modeling tools

and ANN libraries. The CoNM tool coordinates them and carries out NPO tasks. Since the interplay of various systems is harmonized by the NEA designed, and NPO-based changes are directly projected on NMDL models, any kind of improvement can precisely transferred to the original systems. Since furthermore the ANN behavior as well as the knowledge construction become transparent, the NMDL is able to serve as communication platform for human- and machine-based simulation model constructors.

Main Research Question: Interconnecting the insights of the sub-research questions identified, the main research question, that is, (*How can neuronal processes be modeled with help of knowledge modeling description languages?*), can be answered as follows: Neuronal processes can be modeled using an adequate graphical modeling language which provides relevant modeling items and a technical substantiation of common process modeling entities with ANN mechanisms. An NPM addresses the visualization of any kind of system. SEKO addresses the identification and collection of knowledge. NPS addresses the scenario-based simulation of models constructed so that the temporal effects of model sequences can be visualized. The NPO addresses the data-driven modification of process models constructed so that the current process model as well as the current knowledge base can be changed. Since the NMDL serves for both, the specification of ANN-based operations as well as the visualization of corresponding results, it functions as a communication platform for human- as well as machine-based model constructors.

Since the term *modeling* implies changes within the models constructed, the management of the neuronal and human knowledge base is realized through the *CoNM Procedure*, and the symbiotic *NKMM* is enabled. Learning procedures in this context initially cares about the approximation of artificial, ANN-based systems to real systems, which can be found in reality. When approximations have been crystallized, learning procedures can care about the optimization of process networks, processes, tasks, activities and activations.

Through CoNM implementation, the corresponding CoNM tasks can be specified and initiated using modeling tools such as the *Modelangelo*. With the aid of ANN libraries such as *PyBrain*, the CoNM tool carries out CoNM tasks. By following the NEA designed, the bidirectional modeling in AR, VR, MR, XR and reality has been enabled. This has been realized through the CoNM implementation.

Hence, considering the sub-artifacts created, it can be concluded that the objectives issued in section 1.2 have been fulfilled and the *Concept of Neuronal Modeling* (CoNM) can be provided as the main artifact.

8.2.2 Methodology

As was worked out using the methodological approach in section 3.3 (see Fig. 3.8), this is a design-oriented research. Hence, the classical phases of an adequate research process have been realized.

Problem Statement: The problem statement and its problem analysis are based on the consideration of contemporary research. Particularly, the analysis of state-of-the-art concepts of relevant domains regarding the neuronal process modeling has been relevant. So, the foundation for the reuse of tools has been prepared. Further, the strengths and weaknesses have been identified tool-wise. The research gap has been made transparent by the benchmark of tools over relevant domains using the CoNM selection method. The very same research gap has been confirmed by an SLR.

Further, it was identified that a joint terminology for the *ANN Process Domain* is missing. By the SLR, domain-specific concepts and definitions have been identified. These have been the foundation for the harmonized fusion of isolated domains and the *ANN Process Domain*. Based on this understanding, the CoNM has been built.

Objective Analysis: The objective analysis has lead to a CoNM terminology and the specification of the *ANN Process Domain* (see Fig. 3.1). Particularly, the differentiation of the *Neuronal Process Modeling* (NPM), the *Neuronal Process Simulation* (NPS) as well as the *Neuronal Process Optimization* (NPO) has been widely discussed at conferences (Gronau and Grum, 2017) as well as workshops and courses the author has held at the University of Potsdam. Examples can be found from the seminar called *ANN in Business Informatics*, at *Knowledge Management Exercises* and at *Business Process Management Exercises*. The validation of requirements derived from objectives have been discussed with experts from the domain of knowledge management, process modeling and optimization, simulation and AI, and they have been optimized over the course of three years. They have guided the CoNM implementation.

Design and Implementation: The realization of sub-artifacts is based on the following two things: First, the conceptional design that has partly been presented at conferences, for instance, the beginnings of the NMDL and the meta-model (Gronau and Grum, 2017; Grum and Gronau, 2018a), as well as the symbiotic KM (Grum, 2020); further sub-artifacts have been discussed at the half-year colloquium at the University of Potsdam.

Second, the implementation of the artifacts conceptualized; CoNM software prototypes such as the *Augmentor* have been discussed, for instance, stance at conferences (Grum and Gronau, 2017; Grum and Gronau, 2018b) and at workshops held by the author before the empirical evaluation.

Demonstration: The application of CoNM tools has been realized through experiments and empirical research. Each experiment has been created by following an individual CoNM proceeding cycle by the author. Experiences generated by the realization of CoNM activities have been considered at the software prototypes.

Evaluation: The evaluation of relevant objects of investigation as well as the CoNM constructed has been carried out by the author as well as by experts from the domain of knowledge management, business process management, machine learning and ANN. Since the majority of evaluation criteria has been identified in literature, the connectivity to contemporary approaches has been guaranteed. Since the practical applicability of the CoNM has been evaluated on behalf of a simulator feedbacked with SMEs, a reality check was realized.

Communication: The final communication of this research is planned to be directed to the following group of persons: 1) experts from relevant domains as well as the common research community, by whom the scientific discourse shall be initiated; 2) practitioners to evaluate the CoNM potential progress economy. Here, the relevant roles of the NKMM will be focused (cf. section 4.5) upon to ensure that any kind of stakeholder will be profited from CoNM insights. Research artifacts being communicated do not only refer to this written form of research, but they also include the public CoNM repository and presentations, workshops and seminars about this research.

DSR Guidelines: In accordance with the design-science research guidelines of Hevner et al. (2004), this contribution satisfies the requirements for effective design-science research and is complete, as indicated in Table 8.1.

8.2.3 Overall Result

Having had the intention to present a harmonized concept of a neuronal modeling, this research has presented the least required number of sub-artifacts. This is evident from the complex interconnections between CoNM sub-artifacts, as shown in Fig. 8.1. Each kind of connection is justified in the following:

Table 8.1 Design-Science Research Guidelines.

Guideline	Description
Guideline 1: Design as an artifact	The author designs a flexible modeling concept described by a modeling language and its mathematical formulation and connected to a technical substantiation. Both frameworks are implemented as a computational model. A CoNM tool is designed that carries out novel modeling (NPM), novel simulations (NPS) and novel optimizations (NPO).
Guideline 2: Problem relevance	Considering the previously mentioned artifacts, the business problem of having inefficient business processes, imprecise process analyses and simulations as well as non-transparent artificial neuronal network models is overcome by an easy-to-use modeling interface. A common proceeding is presented that can serve for structured realization of CoNM tasks and be applied to different analytical infrastructures. As the framework is interrelated with a symbiotic knowledge management model, the knowledge-oriented development of artificial neuronal network as well as the management of human- and machine-based knowledge carriers are reasonable, given a highly evolutionary environment and continual application of the framework.
Guideline 3: Design evaluation	The efficacy of the designed artifacts was demonstrated rigorously through experiments. The utility and quality of the CoNM tool performance was demonstrated using various training and test runs. The execution precisely followed the documented mathematical expressions and novel learning principles designed. Therefore, validation of the theoretical model and simulation are valid within a theoretical and small setting, and will be transferred to a larger example in a real-life setting as the next step.
Guideline 4: Research contributions	The design-science contributions of this research are the proposed CoNM along with its sub-artifacts and evaluation results in the form of models constructed, simulations and optimizations carried out and performance analysis. These contributions advance our understanding of the manner in which to optimize any form of processes within the modern dynamic artificial neuronal network infrastructures. Regarding the related works of Tab. 2.3, this integrates all the related works, satisfies categories identified here and contributes to close the research gap.
Guideline 5: Research rigor	Research on approximation using artificial neuronal network strategies has long been based on complex algorithms such as backpropagation and backpropagation through time (Schmidhuber, 2015). In this contribution, actual entities provide the underlying approximation task on an arbitrary level of detail, which allows for efficient combination of pre-trained artificial neuronal networks and enables the development of more complex process representations from neuron to activation, activity, task, process, process network and scenario-based use of process networks.

(Continued)

Table 8.1 (Continued)

Guideline	Description
Guideline 6: Design as a search process	As discussed previously, the implementation of neuronal modeling strategies, application and benchmarking in artificial neuronal network based environments in iterations is essential. The author studied variations in realization strategies over a period of five years within the aforementioned experiments. Creativity and problem-solving capabilities were involved in the construction of a CoNM.
Guideline 7: Communication of research	The presentation of this research is aimed at an audience familiar with the process and knowledge management theory and artificial neuronal networks. Even so, the contribution provides useful information for managerial audiences. While the author presents a thorough discussion of economic performance criteria, the contribution provides evidence for both technical implementations and economic reasoning.

- 1. Interconnections with Experiments:** Since experiments are required for the proof-of-concept, experiments presented here have focused on the demonstration of all the other CoNM sub-artifacts.
- 2. Interconnections with Implementation:** Since experiments were intended to be realized and not to be conceptualized only, the CoNM has been implemented. This included the implementation of other CoNM sub-artifacts, because of which these experiments could be carried out with the aid of the software tools.
- 3. Interconnections with NEA:** Since the process of modeling has been identified to include various kinds of experts and stakeholders, and further CoNM sub-artifacts demand for complex IT infrastructure component interactions, a neuronal enterprise architecture has been developed. This structures any kind of architectural requirement and, in this case, has prepared the CoNM tool implementation.
- 4. Interconnections with Proceeding:** Since the CoNM demands for a complex sequence of activities, phases have been identified which enable the structured and target-oriented CoNM project realization. Arranged by a harmonized proceeding model, it becomes clear when a certain CoNM sub-artifact needs to be used. Further, it becomes clear as to which kind of output can be expected from the phase-specific use of CoNM tools.
- 5. Interconnections with NKMM:** The kind of activities and by whom these shall be carried out has been structured by the symbiotic knowledge management

model called NKMM. These activities consider the use of CoNM concepts. Particularly, by the cooperative modeling of human and machine-based model constructors, the management of human and ANN-based knowledge carriers is enabled.

6. **Interconnections with NMDL:** Since the CoNM applies various concepts, the NMDL serves as a communication platform for human- as well as machine-based model constructors. Using the NMDL, the CoNM tool input is specified and CoNM output is visualized. So, using the NMDL, the use of other CoNM sub-artifacts can be specified and visualized as well.
7. **Interconnections with SEKO:** Since SEKO refers to novel CoNM concepts for the identification of knowledge objects, these have been implemented in CoNM tools and demonstrated through experiments. The NEA structures the SEKO data flows; its results and initialization are visualized by the NMDL.
8. **Interconnections with Learning Principles:** Since CoNM learning principles refer to novel training procedures being realized within the *ANN Process Domain*, the novel procedures are demonstrated by experiments and with the aid of implemented CoNM tools. Again, the NEA structures the corresponding data flows; the results and initialization are visualized by the NMDL.

On the one hand, the intention to present a harmonized concept for neuronal modeling was through the complexity and extent of this contribution. This includes the limitation of some ANN architectures in the first CoNM version. For instance, alternatively, the focus of this contribution could have been the NPM only. However, a limited focus and the disregarding of one of the sub-artifacts presented would have lead to a reduced effectiveness, comprehensibility and capacity of the CoNM. Moreover, various connections for consecutive research would not have become transparent at all.

Hence, compared to Tab. 2.3, the major contribution lies in the integration of various domain-specific knowledge bases into one coherent modeling concept. This provides a basis for the transparent construction of process models, knowledge models, simulation models, optimization models and ANN models; it also provides the basis for their technical substantiation. Through its application in experiments, CoNM has been proven to be a modeling concept. Using the CoNM tool, helpful guidelines regarding process optimization can be derived for both managers and technical experts by all human- and machine-based model constructors.

8.2.4 Single Results

Although the construction of a CoNM has been the primary focus of this contribution, a collection of sub-artifacts was required in order to position the CoNM reasonable and realize the CoNM at all. Regarding the objectives presented in section 1.2, one can appraise the results of individual sub-artifacts critically.

1. **NDML:** A knowledge modeling and description language has been realized that maps to ANN and models the working of brains. Therefore, it serves as a communication platform for human- as well as machine-based model constructors.
2. **CoNM Proceeding:** A proceeding has been realized that guides the CoNM realization. In twelve phases, the complex sequence of activities has been structured.
3. **NKMM:** An ordering system has been realized that enables the measure-based evolution of ANN systems and model systems designed, thereby enabling the symbiotic knowledge management of human- as well as machine-based knowledge carriers.
4. **SEKO:** A systematic approach to exhaust knowledge objects based on relevant data has been realized. So, structural as well as dynamic signal-based knowledge can be extracted from ANN systems.
5. **Learning Principles:** The CoNM learning principles have been formulated in a way that the architecture of ANN could be interpreted such that it refers to real processes. So, the construction of ANN-based process models, simulations and optimizations was enabled
6. **Experiments:** In experiments, ANN have been trained to demonstrate the identification of tacit knowledge. Further, ANN have been trained to demonstrate the simulation of knowledge use. Lastly, ANN have been trained to demonstrate the optimization of the neuronal knowledge base; this has lead to a process improvement.
7. **NEA:** An architectural design for the CoNM has been realized which structures the interplay of CoNM tools.
8. **Implementation:** An implementation of CoNM software and hardware has been realized as a proof-of-concept. These have been demonstrated to function by conducting experiments. Furthermore, the CoNM has been proven to be applicable to a realistic setting, which was shown by a LEGO-based simulator.

These single results shown that the CoNM objectives have been realized. Jointly, they have served as an integral part of the main artifact of the CoNM.

8.3 Outlook

Through the versatile dealing with the concept of neuronal modeling, the following numerous steps have been identified, which address their potential for ongoing research.

Professionalization of the CoNM Tools: Because of the intention to demonstrate the functioning of CoNM tools designed, these have been set up as prototypes. Therefore, the following points are needed to professionalize the CoNM tools presented in this contribution.

- 1) The intended usability concept allows the controlling of the CoNM software by just some buttons using the modeling tool. For instance, this allows the start of a training run when the *ActivationView* is active in the modeling tool and holds the activity to be trained. Alternatively, the same training run can be started when the same activity item had been highlighted in the *ActivationOverview*, the *ProcessANNOverview*, the *NeuronalOverview*, the *InformationalOverview*, the *OrganizationalOverview* or any kind of related view. For the demonstration cases, only one variant was implemented.

Another example refers to easy navigation through models. Although *Modelangelo* technically supports the creation of referenced modeling items, and changes in one referenced item affects any other referenced item throughout the views, the usability characteristics of the NMDL have not been realized yet. These are shown by the white squares in Fig. 4.17 and would allow to use those interlinked modeling items for navigating throughout the model hierarchies for example.

- 2) The CoNM holds the potential for an auto-completion of models constructed. For instance, knowledge objects identified by SEKO concepts can be stored automatically in the *Knowledge Object Overview*, the *Information Object Overview*, and the related *Activation Views* and *Activity Views*. Further, behavioral changes of NPO procedures can automatically result in the corresponding *Process Views* and the *Network Views*. In general, this supports the construction of a consistent model base in the sense of an object repository.
- 3) The detection of complex cyclic graphs need to be considered so that complex recurrent connections can be identified by the CoNM automatically. For carefree, daily working with the CoNM, this will be essential.
- 4) Prototypes have to be progressed so that they can realize a bidirectional modeling in XR (see multiple-device-mode at Fig. 5.1). This would need an adequate online event management system that cares about simultaneous modifications of different model constructors being in conflict.

- 5) As was demanded by the NEA set up (see Fig. 4.8), a versatile right management system is required. Particularly, in the long run, if various customers are intended to operate on a public CoNM system, security issues will need to be addressed.
- 6) Although the integration of third-party systems via interfaces has been demonstrated, industry standards demand for more than a database or file integration. If this is required, the customer's individual needs for a specific third-party system integration will have to be considered. Subsequent demonstration cases should further address the interplay with process planning and controlling systems (see Fig. 4.8).
- 7) Although the CoNM was demonstrated by including the modeling tool called *Modelangelo* and the library called *PyBrain*, the CoNM and its concepts are not dependent on them. Hence, the option to base models on powerful libraries such as *TensorFlow* or alternative modeling tools such as *ARIS Platform* or rather open source alternatives is attractive. This particularly holds a potential if strengths of different tools are integrated by the CoNM tools. Subsequent demonstration cases should further address the operation in different technical environments (see Fig. 4.8).
- 8) Further points to professionalize the CoNM tools can be derived from the criteria being issued by the selection tool. Beside the points presented above, here, one can find, for instance, the item-specific version control, the speed-optimized processing or a life cycle management.

Extension of the CoNM Model Base: Although a collection of the most relevant ANN architectures have been modeled, the collection of models should be extended. In the case that these are interpreted as reference models (cf. Def. 20), a company's contemporary collection of processes can be improved by considering them: Since ANN architectures, process models, simulation models and optimization models of the NMDL can be provided by the *Model Warehouse*, ANN reference architectures can be imported easily. Further, the *Simulation Warehouse* and the *Optimization Warehouse* should be extended further by NPS or rather NPO results so that the behavior of models that have been provided by the *Model Warehouse* becomes clear.

Since the *Planning Warehouse*, the *Workflow Warehouse* and the *Application System Warehouse* will majorly refer to customer-specific configurations, these kinds of warehouses at least should provide examples of the reference architecture. This will simplify the set up of a CoNM configuration.

Extension of the CoNM Concept Base: This contribution explains that the set of concepts for the CoNM is not closed, implying that the CoNM holds the potential to

be extended by concepts for the NPM, NPS and NPO. This includes the consideration of concepts of neighboring domains in the *ANN Process Domain*.

In case the concepts of neighboring domains are considered for the modeling concept of CoNM, the need exists to examine the extent to which they contribute to the neuronal modeling. Further, an analysis will be needed to see if novel methods are in concurrence with existing methods, so that the strengths and weaknesses of the methods become clear. This includes for instance the transfer of free learning tasks (Def. 22) as well as reinforced learning tasks (Def. 23). The established CoNM has so far only dealt with fixed learning tasks (Def. 21).

In case novel concepts are constructed in the *ANN Process Domain*, they might have a potential for the *Process Domain* or the *ANN Domain*. Examples can be found in the tempo-spatial positioning of modeling items. Although this has been considered based on the modeling concept designed, an optimization because of their spatial positioning has not been demonstrated yet. Moreover, knowledge objects can be based on fuzzy knowledge interpretation, which are in accordance with fuzzy sets (Def. 25) and fuzzy systems (Def. 28).

Extension of the CoNM Experiment Base: Since the experiments provided by this contribution are limited to the functional proof of novel concepts developed, the base of experiments needs to be extended. Although the CoNM has been proven to be applicable in real settings such as a LEGO-based production system, a focus shall lie on the verification of CoNM concepts using more complex datasets, ANN architectures, scenarios, simulation systems, third-party products etc. so that the validation level and generalization capabilities of ANN can be increased. If the experiment base and the different kinds of CoNM warehouses are extended systematically, the CoNM can hold the potential to create more complex knowledge bases.

Expansion of the Fields of Application: Inspired by software development, the CoNM holds further the potential for debugging ANN, since knowledge and behavior of ANN can be visualized and compared with an intended state. In addition to the application of the CoNM to production and business processes, an application to physical, chemical and astronomical processes is very promising, since on the one hand data is available in a large amount and on the other hand new insights about the world and interwoven reality could be gained by analyzing and simulating the resulting ANN models.

Hopefully, these will help to comprehend how the world holds together (concrecence) and becomes real in public (satisfaction).

Bibliography

- Aalst, W.M.P. van der (1999). "Formalization and verification of event-driven process chains". In: *Information and Software Technology* 41.10, pp. 639–650. ISSN: 0950-5849. DOI: [https://doi.org/10.1016/S0950-5849\(99\)00016-6](https://doi.org/10.1016/S0950-5849(99)00016-6). URL: <http://www.sciencedirect.com/science/article/pii/S0950584999000166>.
- Aalst, W.M.P. van der et al. (2003). "Workflow Patterns". In: *Distributed and Parallel Databases* 14.1, pp. 5–51. ISSN: 1573-7578. DOI: <https://doi.org/10.1023/A:1022883727209>. URL: <https://doi.org/10.1023/A:1022883727209>.
- Abadi, Martín et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: <https://www.tensorflow.org/>.
- Abadi, Martín et al. (2016). "TensorFlow: A System for Large-scale Machine Learning". In: *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*. OSDI'16. Savannah, GA, USA: USENIX Association, pp. 265–283. ISBN: 978-1-931971-33-1. URL: <http://dl.acm.org/citation.cfm?id=3026877.3026899>.
- Ackley, David H., Geoffrey E. Hinton, and Terrence J. Sejnowski (1985). "A Learning Algorithm for Boltzmann Machines". In: *Cognitive Science* 9.1, pp. 147–169. DOI: https://doi.org/10.1207/s15516709cog0901_7. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1207/s15516709cog0901_7. URL: https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog0901_7.
- Adam, Sebastian et al. (2013). *Studie—BPM Suites 2013*. Report. Fraunhofer IESE.
- Adam, Sebastian et al. (2014). *Business Process Management—Marktanalyse 2014. BPM Suites im Test*. Report IESE-Report 072.14/D. Fraunhofer IESE.
- Adriaans, W. and J.T. Hoogakker (1989). "Planning an information system at Netherlands gas". In: *Long Range Planning* 22.3, pp. 64–74. ISSN: 0024-6301. DOI: [https://doi.org/10.1016/0024-6301\(89\)90008-3](https://doi.org/10.1016/0024-6301(89)90008-3). URL: <http://www.sciencedirect.com/science/article/pii/0024630189900083>.
- Aggarwal, Charu C. (2014). *Data Classification: Algorithms and Applications*. 1st. Chapman & Hall/CRC. ISBN: 1466586745, 9781466586741.
- Albers, Albert et al. (2014). "Knowledge management in product generation development—An empirical study". In: *25th Symposium Design for X, DFX 2014, Bamberg; Germany, 1. October 2014 through 2. October 2014. Ed.: D. Krause*. TuTech-Verl., Berlin, pp. 13–24. ISBN: 978-394149279-0.

- Alcamo, I.E. and J. Bergdahl (2003). *Anatomy Coloring Workbook*. Anatomy Coloring Workbook. Random House. ISBN: 9780375763427. URL: <https://books.google.de/books?id=NiTLf7gIn04C>.
- Alexander, Allen T. and Stephen J. Childe (2012). "A Framework for the Transfer of Knowledge between Universities and Industry". In: *Advances in Production Management Systems. Value Networks: Innovation, Technologies, and Management*. Ed. by Jan Frick and Bjørge Timenes Laugen. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 534–548. ISBN: 978-3-642-33980-6.
- Allee, V. (1997). *The Knowledge Evolution: Expanding Organizational Intelligence*. Business Briefcase Series. Butterworth-Heinemann. ISBN: 9780750698429. URL: <https://books.google.hu/books?id=Gjb9GttdMIwC>.
- Allweyer, T. (1998). "Modellbasiertes Wissensmanagement". In: *Information Management* 13.1, pp. 37–45.
- Allweyer, T. (2005). *Geschäftsprozessmanagement*. W3I GmbH 1. Auflage Herdecke Bochum.
- Allweyer, T. and A.-W. Scheer (1995). "Modellierung und Gestaltung adaptiver Geschäftsprozesse". In: *IWi-Heft 115*. Saarbrücken, Germany. In German.: Institut für Wirtschaftsinformatik, Universität Saarbrücken.
- Alpar, P. et al. (2002). *Anwendungsorientierte Wirtschaftsinformatik: Eine Einführung in die strategische Planung, Entwicklung und Nutzung von Informations- und Kommunikationssystemen*. 3. Braunschweig: Vieweg.
- Althöfer, Ingo and Klaus-Uwe Koschnick (1991). "On the convergence of "Threshold Accepting"". In: *Applied Mathematics and Optimization* 24.1, pp. 183–195. ISSN: 1432-0606. DOI: <https://doi.org/10.1007/BF01447741>. URL: <https://doi.org/10.1007/BF01447741>.
- American Society of Mechanical Engineers (1947). *A.S.M.E. standard operation and flow process charts*. The American society of mechanical engineers. URL: <https://books.google.de/books?id=hBAhAAAAMAAJ>.
- AMICE, ESPRIT Consortium (1993). *CIMOSA: open system architecture for CIM*. Research reports ESPRIT: AMICE. Springer-Verlag. ISBN: 9783540562566. URL: <https://books.google.de/books?id=oNFhngEACAAJ>.
- Ammon, Danny et al. (2009). "Management of Knowledge-Intensive Healthcare Processes on the Example of General Medical Documentation". In: *Business Process Management Workshops*. Ed. by Danilo Ardagna, Massimo Mecella, and Jian Yang. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 324–335. ISBN: 978-3-642-00328-8.
- Angelopoulou, Anastassia et al. (2005). "Automatic Landmarking of 2D Medical Shapes Using the Growing Neural Gas Network". In: *Computer Vision for Biomedical Image Applications*. Ed. by Yanxi Liu, Tianzi Jiang, and Changshui Zhang. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 210–219. ISBN: 978-3540-32125-5.
- Angermann, A. (1963). "Betriebsführung und Operations Research". In: *Vorträge des Verbandes der Hochschullehrer für Betriebswirtschaft e.V. auf seiner Jahrestagung 1962 vom 12. bis 16. Juni 1962 in Hamburg*. Frankfurt am Main: Nowack.
- Angermann, A. et al. (2014). *MATLAB—Simulink—Stateflow: Grundlagen, Toolboxen, Beispiele*. De Gruyter. ISBN: 9783486859102. URL: <https://books.google.de/books?id=Kn3pBQAAQBAJ>.

- Argyris, C. (1990). *Overcoming Organizational Defenses: Facilitating Organizational Learning*. Allyn and Bacon. ISBN: 9780205123384. URL: <https://books.google.de/books?id=7i3AAAAIAAJ>.
- Argyris, C. and D.A. Schön (1978). *Organizational learning: a theory of action perspective*. Addison-Wesley series on organization development. Addison-Wesley Pub. Co. URL: <https://books.google.de/books?id=VYDIXAEACAAJ>.
- Argyris, C. and D.A. Schön (1996). *Organizational Learning II: Theory, Method, and Practice*. Addison-Wesley OD series Bd. 2. Addison-Wesley Publishing Company. ISBN: 9780201629835. URL: <https://books.google.de/books?id=3aMoAQAAQAAJ>.
- Argyris, Chris (2003). "A Life Full of Learning". In: *Organization Studies* 24.7, pp. 1178–1192. DOI: <https://doi.org/10.1177/01708406030247009>. eprint: <https://doi.org/10.1177/01708406030247009>. URL: <https://doi.org/10.1177/01708406030247009>.
- Armbrust, Michael et al. (2015). "Spark SQL: Relational Data Processing in Spark". In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. SIGMOD '15. Melbourne, Victoria, Australia: ACM, pp. 1383–1394. ISBN: 978-1-4503-2758-9. DOI: <https://doi.org/10.1145/2723372.2742797>. URL: <http://doi.acm.org/10.1145/2723372.2742797>.
- Armstrong, D. M. (1973). *Belief, Truth and Knowledge*. Cambridge University Press.
- Ashby, W.R. (1960). *Design for a brain: the origin of adaptive behaviour*. Science paperbacks. Wiley. URL: <https://books.google.de/books?id=CSAMAAAAMAAJ>.
- Aydt, H. et al. (2009a). "Symbiotic Simulation Systems: An Extended Definition Motivated by Symbiosis in Biology". In: *Proceedings of the 22nd Workshop on Principles of Advanced and Distributed Simulation*. Rome, Italy, pp. 109–116.
- Aydt, Heiko et al. (2009b). "Research Issues in Symbiotic Simulation". In: *Winter Simulation Conference*. WSC '09. Austin, Texas: Winter Simulation Conference, pp. 1213–1222. ISBN: 978-1-4244-5771-7. URL: <http://dl.acm.org/citation.cfm?id=1995456.1995624>.
- Azevedo, Frederico A.C. et al. (2009). "Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain". In: *The Journal of Comparative Neurology* 513.5, pp. 532–541. ISSN: 1096-9861. DOI: <https://doi.org/10.1002/cne.21974>. URL: <http://dx.doi.org/10.1002/cne.21974>.
- Babulak, Eduard and Ming Wang (2010). "Discrete Event Simulation: State of the Art". In: *Discrete Event Simulations*. Ed. by Aitor Goti. Rijeka: IntechOpen. Chap. 1. DOI: <https://doi.org/10.5772/9894>. URL: <https://doi.org/10.5772/9894>.
- Bach, Sebastian et al. (July 2015). "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation". In: *PLOS ONE* 10.7, pp. 1–46. DOI: <https://doi.org/10.1371/journal.pone.0130140>. URL: <https://doi.org/10.1371/journal.pone.0130140>.
- Bach, V., H. Österle, and P. Vogler (2000). *Business Knowledge Management in der Praxis: Prozessorientierte Lösungen zwischen Knowledge Portal und Kompetenzmanagement*. Springer, Berlin Heidelberg.
- Back, T. (1996). *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press. ISBN: 9780195356700. URL: <https://books.google.de/books?id=EaN7kv15coYC>.
- Back, T., U. Hammel, and H. . Schwefel (1997). "Evolutionary computation: comments on the history and current state". In: *IEEE Transactions on Evolutionary Computation* 1.1, pp. 3–17. ISSN: 1089-778X. DOI: <https://doi.org/10.1109/4235.585888>.

- Backhaus, Klaus et al. (2008). *Multivariate Analysemethoden: Eine anwendungsorientierte Einführung*. 12., vollständig überarbeitete Auflage. Springer, Berlin. ISBN: 3540850449. URL: <http://www.amazon.de/Multivariate-Analysemethoden-Eine-anwendungsorientierte-Einf%C3%BChrung/dp/3540850449%3FSubscriptionId%3D13CT5CVB80YFWJEPWS02%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D3540850449>.
- Baddeley, A (1986). *Oxford psychology series, No. 11. Working memory*. New York, NY, US.
- Baddeley, A J (2000). "The episodic buffer: a new component of working memory?" In: *Trends in Cognitive Sciences* 4, pp. 417–423.
- Baddeley, Alan (Oct. 2003). "Working memory: looking back and looking forward." In: *Nature reviews. Neuroscience* 4.10, pp. 829–839. ISSN: 1471-003X. DOI: <https://doi.org/10.1038/nrn1201>. URL: <http://dx.doi.org/10.1038/nrn1201>.
- Baddeley, Alan (2012). "Working Memory: Theories, Models, and Controversies". In: *Annual Review of Psychology* 63.1. PMID: 21961947, pp. 1–29. DOI: <https://doi.org/10.1146/annurev-psych-120710-100422>. eprint: <https://doi.org/10.1146/annurev-psych-120710-100422>. URL: <https://doi.org/10.1146/annurev-psych-120710-100422>.
- Baddeley, Alan, Susan Gathercole, and Costanza Papagno (1998). "The phonological loop as a language learning device." In: *Psychological review* 105.1, p. 158.
- Baddeley, Alan D. (2002). "Is Working Memory Still Working?" In: *European Psychologist* 7.2, pp. 85–97. DOI: <https://doi.org/10.1027/1016-9040.7.2.85>. eprint: <https://doi.org/10.1027/1016-9040.7.2.85>.
- Baeck, T., D.B. Fogel, and Z. Michalewicz (1997). *Handbook of Evolutionary Computation*. Taylor & Francis. ISBN: 9780750308953. URL: <https://books.google.de/books?id=n5nulZvmpAC>.
- Baehrens, David et al. (2010). "How to explain individual classification decisions". In: *Journal of Machine Learning Research* 11.Jun, pp. 1803–1831.
- Baetge, J. (1974). *Betriebswirtschaftliche Systemtheorie: regelungstheoretische Planungs-Überwachungsmodelle für Produktion, Lagerung und Absatz*. Moderne Lehrtexte / Wirtschaftswissenschaften. Westdeutscher Verlag. ISBN: 9783531111773. URL: <https://books.google.de/books?id=2wcSPwAACAAJ>.
- BAIR, Berkeley AI Research (Nov. 2018a). *Caffe 2 Repository*. URL: <https://github.com/pytorch/pytorch> (visited on 11/03/2018).
- BAIR, Berkeley AI Research (Nov. 2018b). *Caffe 2 Webpage*. URL: <https://caffe2.ai/> (visited on 11/03/2018).
- BAIR, Berkeley AI Research (Nov. 2018c). *Caffe Repository*. URL: <https://github.com/BVLC/caffe/> (visited on 11/03/2018).
- BAIR, Berkeley AI Research (Nov. 2018d). *Caffe Webpage*. URL: <http://caffe.berkeleyvision.org/> (visited on 11/03/2018).
- Balzert, H. (1999). *Lehrbuch der Objektmodellierung: Analyse und Entwurf ; mit CD-ROM*. Lehrbücher der Informatik. Spektrum Akademischer Verlag, XVIII, 558 S. ISBN: 978-3-8274-1705-3. URL: http://aleph.bib.uni-mannheim.de/F/?func=find-b&request=308066812&find_code=020&adjacent=N&local_base=MAN01PUBLIC&x=0&y=0.
- Balzert, Helmut (2009). *Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering*. 3., Aufl. Heidelberg, Neckar: Spektrum Akademischer Verlag, XVIII, 558 S. ISBN: 978-3-8274-1705-3. URL: http://aleph.bib.uni-mannheim.de/F/?func=find-b&request=308066812&find_code=020&adjacent=N&local_base=MAN01PUBLIC&x=0&y=0.

- Bamberg, G. and A. G. Coenenberg (1974). *Betriebswirtschaftliche Entscheidungslehre*. München: Vahlen.
- Banks, J. (1998). *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*. A Wiley-Interscience publication. Wiley. ISBN: 9780471134039. URL: <https://books.google.de/books?id=dMZ1j3TBgAC>.
- Banks, Jerry et al. (2005). *Discrete-Event System Simulation*. 4th ed. Prentice Hall.
- Barlow, H.B. (1989). "Unsupervised Learning". In: *Neural Computation* 1.3, pp. 295–311. DOI: <https://doi.org/10.1162/neco.1989.1.3.295>. eprint: <https://doi.org/10.1162/neco.1989.1.3.295>.
- Barry, Robert J et al. (2004). "EEG differences in children as a function of resting-state arousal level". In: *Clinical Neurophysiology* 115.2, pp. 402–408.
- Barry, Robert J et al. (2007). "EEG differences between eyes-closed and eyes-open resting conditions". In: *Clinical Neurophysiology* 118.12, pp. 2765–2773.
- Bashe, Charles J. et al., eds. (1986). *IBM's Early Computers*. Cambridge, MA, USA: MIT Press. ISBN: 0-26202225-7.
- Basheer, I.A and M Hajmeer (2000). "Artificial neural networks: fundamentals, computing, design, and application". In: *Journal of Microbiological Methods* 43.1. Neural Computing in Microbiology, pp. 3–31. ISSN: 0167-7012. DOI: [http://dx.doi.org/10.1016/S0167-7012\(00\)00201-3](http://dx.doi.org/10.1016/S0167-7012(00)00201-3). URL: <http://www.sciencedirect.com/science/article/pii/S0167701200002013>.
- Bastien, F. et al. (Nov. 2012). "Theano: new features and speed improvements". In: *ArXiv e-print*. arXiv: 1211.5590 [cs.SC].
- Bateson, G. (1972). *Steps to an ecology of mind*. Ballantine Books. URL: <https://books.google.de/books?id=IcG0AAAAIAAJ>.
- Bayer, J. et al. (Nov. 2018). *PyBrain Webpage*. URL: <http://pybrain.org/> (visited on 11/03/2018).
- Becker, F.G. (2002). *Lexikon des Personalmanagements: über 1000 Begriffe zu Instrumenten, Methoden und rechtlichen Grundlagen betrieblicher Personalarbeit*. Dtv: Beck-Wirtschaftsberater. Dt. Taschenbuch-Verlag. URL: <https://books.google.de/books?id=BZpCLgAACAAJ>.
- Becker, J., L. Ehlers, and R. Schütte (1998). "Grundsätze ordnungsgemäßer Modellierung—Konzeption, Vorgehensmodell, technische Realisierung, Nutzen". In: *Tagungsband zur Statustagung des BMF. Softwaretechnologie, Bonn. In German*.
- Becker, J., M. Rosemann, and R. Schütte (1995). "Grundsätze ordnungsgemäßer Modellierung". In: *Wirtschaftsinformatik* 37.5, pp. 435–445.
- Becker, J. and R. Schütte (1996). *Handelsinformationssysteme*. Landsberg, Lech: Moderne Industrie.
- Becker, J. and R. Schütte (2004). *Handelsinformationssysteme*. Redline Wirtschaft bei moderne industrie. mi-Wirtschaftsbuch. ISBN: 9783636031440. URL: <https://books.google.de/books?id=LbXrCQ-aIcUC>.
- Becker, J et al. (2000). *Grundsätze ordnungsmäßiger Modellierung (GoM) / Westfälische Wilhelms-Universität Münster-Institut für Wirtschaftsinformatik, IDS Scheer AG, Josef Friedr. Tech. rep.* Bremke & Hoerster GmbH & Co.
- Becker, Jörg (2010). "Gestaltungsorientierte Wirtschaftsinformatik: Ein Plädoyer für Rigor und Relevanz". In: ed. by Hubert Österle, Robert Winter, and Walter Brenner. Eigenverlag. Chap. Prozess der gestaltungsorientierten Wirtschaftsinformatik, pp. 13–17.

- Becker, Jörg, Patrick Delfmann, and Ralf Knackstedt (2004). "Adaption fachkonzeptioneller Referenzprozessmodelle". In: *Industrie Management* 20.1, pp. 19–22.
- Becker, Jörg and Ralf Knackstedt (2004). "Referenzmodellierung im Data-Warehousing—State-of-the-Art und konfigurative Ansätze für die Fachkonzeption". In: *Wirtschaftsinformatik* 46.1, pp. 39–49. ISSN: 18618936. DOI: <https://doi.org/10.1007/BF03250994>. URL: <https://doi.org/10.1007/BF03250994>.
- Becker, Jörg and Daniel Pfeiffer (2006). "Beziehungen zwischen behavioristischer und konstruktionsorientierter Forschung in der Wirtschaftsinformatik". In: *Fortschritt in den Wirtschaftswissenschaften: Wissenschaftstheoretische Grundlagen und exemplarische Anwendungen*. Ed. by Stephan Zelewski and Naciye Akca. Wiesbaden: DUV, pp. 1–17. ISBN: 978-3-8350-9199-3. DOI: https://doi.org/10.1007/978-3-8350-9199-3_1. URL: https://doi.org/10.1007/978-3-8350-9199-3_1.
- Beeman, David (2005). "Introduction to Realistic Modeling". In: *Brains, Minds and Media*. Vol. 1. bmm218.
- Beer, S. (1979). *The heart of enterprise*. Managerial cybernetics of organization. Wiley. ISBN: 9780471275992. URL: <https://books.google.de/books?id=1tfAAAAIAAJ>.
- Begleiter, Ron, Ran El-Yaniv, and Golan Yona (Dec. 2004). "On Prediction Using Variable Order Markov Models". In: *J. Artif. Int. Res.* 22.1, pp. 385–421. ISSN: 1076-9757. URL: <http://dl.acm.org/citation.cfm?id=1622487.1622499>.
- Begoña Lloria, M (2008). "A review of the main approaches to knowledge management". In: *Knowledge management research & practice* 6.1, pp. 77–89.
- Bem, D. J. (1995). "Writing a review article for Psychological Bulletin". In: *Psychological Bulletin* 118.2, pp. 172–177.
- Bengio, Y., P. Simard, and P. Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2, pp. 157–166. URL: citeseer.ist.psu.edu/bengio94learning.html.
- Berenji, Hamid R. (Feb. 1992). "A Reinforcement Learning-based Architecture for Fuzzy Logic Control". In: *Int. J. Approx. Reasoning* 6.2, pp. 267–292. ISSN: 0888-613X. DOI: [https://doi.org/10.1016/0888-613X\(92\)90020-Z](https://doi.org/10.1016/0888-613X(92)90020-Z). URL: [http://dx.doi.org/10.1016/0888-613X\(92\)90020-Z](http://dx.doi.org/10.1016/0888-613X(92)90020-Z).
- Berg, Bernd A. (2004). *Markov Chain Monte Carlo Simulations and Their Statistical Analysis*, World Scientific.
- Bergin, Thomas J. (1993). *Computer-Aided Software Engineering: Issues and Trends for the 1990s and Beyond*. Hershey, PA, USA: IGI Global. ISBN: 1878289152.
- Bergstra, James et al. (2010). "Theano: A CPU and GPU math compiler in python". In: *Proceedings of the 9th Python in Science Conference*, pp. 3–10.
- Bernzen, R. (1990). "Modell". In: *Europäische Enzyklopädie zu Philosophie und Wissenschaften*. Ed. by H.J. Sandkuhler and A. Regenbogen, F. Meiner, pp. 425–432. ISBN: 9783787309832. URL: <https://books.google.de/books?id=Q8IMAAAAIAAJ>.
- Bertalanffy, Ludwig v. (1949). "Zu einer allgemeinen Systemlehre." In: *Biologia Generalis*, 19 (1949). S. 114–129.
- Bertalanffy, Ludwig von (1972). "Systemtheorie". In: ed. by R. Kurzrock. Berlin: Colloquium Verlag. Chap. Vorläufer und Begründer der Systemtheorie. S. 17–28.
- Bertrand, Patrice and Melvin F. Janowitz (2002). "Pyramids and weak hierarchies in the ordinal model for clustering". In: *Discrete Applied Mathematics* 122.1-3, pp. 55–81.

- DOI: [https://doi.org/10.1016/S0166-218X\(01\)00354-7](https://doi.org/10.1016/S0166-218X(01)00354-7). URL: [https://doi.org/10.1016/S0166-218X\(01\)00354-7](https://doi.org/10.1016/S0166-218X(01)00354-7).
- Bertsekas, D.P. (1995). *Nonlinear Programming*. Athena Scientific. ISBN: 9781886529144. URL: <https://books.google.de/books?id=QeweAQAAIAAJ>.
- Best, E. (2003). *Geschäftsprozesse optimieren—Der Praxisleitfaden für erfolgreiche Reorganisierung*. Gabler Verlag, Auflage, Wiesbaden.
- Best, Eike (1987). "Structure Theory of Petri Nets: the Free Choice Hiatus". In: *Petri Nets: Central Models and Their Properties*. Ed. by W. Brauer, W. Reisig, and G. Rozenberg. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 168–205. ISBN: 978-3-540-47919-2.
- Bezdek, James C., E.C.-K. Tsao, and Nikhil R. Pal (1992). "Fuzzy Kohonen clustering networks". In: [1992 Proceedings] IEEE International Conference on Fuzzy Systems, pp. 1035–1043.
- Bhardwaj, Kartikeya, Naveen Suda, and Radu Marculescu (2019). "Dream Distillation: A Data-Independent Model Compression Framework". In: *arXiv preprint arXiv: 1905.07072*.
- Bhaskar, Hari Lal and R.P. Singh (2014). "Business Process Reengineering: A Recent Review". In: *GJBM* 8.2, pp. 24–51.
- Biegler, L.T. (2010). *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. MOSSIAM Series on Optimization. Society for Industrial and Applied Mathematics. ISBN: 9780898717020. URL: <https://books.google.de/books?id=ZmIC7w9QnPEC>.
- Bienenstock, EL, LN Cooper, and PW Munro (1982). "Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex". In: *Journal of Neuroscience* 2.1, pp. 32–48. ISSN: 0270-6474. DOI: <https://doi.org/10.1523/JNEUROSCI.02-01-00032.1982>, eprint: <https://www.jneurosci.org/content/2/1/32.full.pdf>. URL: <https://www.jneurosci.org/content/2/1/32>.
- Bishop, C. (1995). "Neural Networks for Pattern Recognition". In: *Oxford University Press, Inc.* P. 1.
- Biskup, J. (1995). *Grundlagen von Informationssystemen*. Vieweg+Teubner Verlag. ISBN: 9783528054946. URL: <https://books.google.de/books?id=cZP0AAAACAAJ>.
- Blasdel, Gary G and Guy Salama (1986). "Voltage-sensitive dyes reveal a modular organization in monkey striate cortex". In: *Nature* 321.6070, p. 579.
- Bloech, J. and G.B. Ihde (1972). *Betriebliche Distributionsplanung: Zur Optimierung der logistischen Prozesse*. Physica-VerlagHD. ISBN: 9783790801095. URL: <https://books.google.de/books?id=PenGAAACAAJ>.
- Bloom, B. S. et al. (1956). *Taxonomy of educational objectives. The classification of educational goals. Handbook I: Cognitive domain*. New York: Longmans Green.
- Böhn, M. (2007). *Prozessmodellierungswerzeuge: 14 Systeme für Dokumentation, Entwurf, Simulation und Analyse im Vergleich*. BARC Software-Evaluation. Oxygon-Verlag. ISBN: 9783937818153. URL: [https://www.tib.eu/de/suchen/id/TIBKAT%3A514229934/Prozessmodellierungswerzeuge-14-Systeme-für-Dokumentation/](https://www.tib.eu/de/suchen/id/TIBKAT%3A514229934/Prozessmodellierungswerzeuge-14-Systeme-f%C3%BCr-Dokumentation/).
- Böhn, M., A. Burkhardt, and M. Gantner (2010). *Prozessmodellierungswerzeuge: Systeme für Dokumentation, Entwurf, Simulation und Analyse im Vergleich ; eine Studie des Business Application Research Center*. BARC Software-Evaluation. B-Eye-Media. ISBN: 9783942201193. URL: <https://books.google.de/books?id=yYfaXwAACAAJ>.

- Böhn, M. et al. (2014). *Prozessmodellierungswerkzeuge: Systeme für Dokumentation, Analyse, Simulation und Prozesssteuerung*. BARC Software-Evaluation. B-Eye-Media. ISBN:9783942201261. URL: <https://books.google.de/books?id=0pGwoAEACAAJ>.
- Boland, N. L. (1996). “A dual-active-set algorithm for positive semi-definite quadratic programming”. In: *Mathematical Programming* 78.1, pp. 1–27. ISSN: 1436-4646. DOI: <https://doi.org/10.1007/BF02614503>. URL: <https://doi.org/10.1007/BF02614503>.
- Bontis, Nick and Alexander Serenko (2009). “A follow-up ranking of academic journals”. In: *Journal of Knowledge Management*.
- Booch, G., J. Rumbaugh, and I. Jacobson (2005). *The Unified Modeling Language User Guide*. Object Technology Series. Addison-Wesley. ISBN: 9780321267979. URL: <https://books.google.de/books?id=BqFQAAAMAAJ>.
- Booch, Grady (1990). *Object Oriented Design with Applications*. Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc. ISBN: 0-8053-0091-0.
- Booch, Grady (1994). *Object-oriented Analysis and Design with Applications (2Nd Ed.)*. Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc. ISBN: 0-8053-5340-2.
- Borshchev, A.V., Y.G. Karpov, and V.V. Roudakov (1997). “COVERS 3.0—An Object-Oriented Environment for Modeling, Simulation and Analysis of Real-Time Concurrent Systems”. In: *Proceedings of the 1st International Workshop on Distributed Interactive Simulation and Real-Time Applications (DIS-RT'97)*. IEEE Computer Society Press.
- Bortz, J. and N. Döring (2006). *Forschungsmethoden und Evaluation für Human- und Sozialwissenschaftler: Limitierte Sonderausgabe*. Springer-Lehrbuch. Springer Berlin Heidelberg. ISBN: 9783540333050. URL: <https://books.google.de/books?id=Ajw6mQEACAAJ>.
- Boschert, Stefan and Roland Rosen (2016). “Digital Twin—The Simulation Aspect”. In: *Mechatronic Futures: Challenges and Solutions for Mechatronic Systems and their Designers*. Ed. by Peter Hehenberger and David Bradley. Cham: Springer International Publishing, pp. 59–74. ISBN: 978-3-319-32156-1. DOI: https://doi.org/10.1007/978-3-319-32156-1_5. URL: https://doi.org/10.1007/978-3-319-32156-1_5.
- Bossel, H. (2004). *Systeme, Dynamik, Simulation: Modellbildung, Analyse und Simulation komplexer Systeme*. Books on Demand. ISBN:9783833409844. URL: <https://books.google.de/books?id=MrSjnxEVmV0C>.
- Bowen, Joe, Padraig Gleeson, and Stephen Larson (Dec. 2018). *Hodgkin Huxley LEMS Tutorial Documentation*. Report. URL: <https://buildmedia.readthedocs.org/media/pdf/hodgkin-huxleytutorial/latest/hodgkin-huxley-tutorial.pdf>.
- Bowen, Joe et al. (Nov. 2019). *LEMS Database—Hodgkin and Huxley—Voltage Gated Potassium Channel*. URL: https://neuroml-db.org/model_info?model_id=NMLCH000116 (visited on 08/26/2019).
- Bower, James M. (1992). “Modeling the nervous system”. In: *Trends in Neurosciences* 15.11, pp. 411–412. ISSN: 0166-2236. DOI: [https://doi.org/10.1016/0166-2236\(92\)90002-P](https://doi.org/10.1016/0166-2236(92)90002-P). URL: <http://www.sciencedirect.com/science/article/pii/016622369290002P>.
- Bower, James M. (1995). “Reverse engineering the nervous system: An *in vivo*, *in vitro*, and *in computo* approach to understanding the mammalian olfactory system”. In: *An Introduction to Neural and Electronic Networks*. Ed. by S.F. Zornetzer, J.L. Davis, and C. Lau. Vol. 2. Academic Press, pp. 3–28.

- Bower, James M. (2005). "Looking for Newton: Realistic Modeling in Modern Biology". In: *Brains, Minds and Media* 2.
- Bower, James M. and David Beeman (1998). *The book of GENESIS—exploring realistic neural models with the GEneral NEural SImulation System* (2. ed.) Springer. ISBN: 978-0-387-94938-3.
- Bower, James M. and David Beeman (2003). *The book of GENESIS—exploring realistic neural models with the GEneral NEural SImulation System (Internet ed.)* Springer. ISBN: 978-0-387-94938-3.
- Bower, James M., Hugo Cornelis, and David Beeman (2013). "GENESIS, The GEneral NEural SImulation System". In: *Encyclopedia of Computational Neuroscience*. Ed. by Dieter Jaeger and Ranu Jung. New York, NY: Springer New York, pp. 1–8. ISBN: 978-1-4614-7320-6. DOI: https://doi.org/10.1007/978-1-4614-7320-ss6_255-1. URL: https://doi.org/10.1007/978-1-4614-7320-6_255-1.
- Boyd, Stephen and Lieven Vandenberghe (2004). *Convex Optimization*. Cambridge University Press. ISBN: 0521833787. URL: <http://www.amazon.com/exec/obidos/redirect?tag=citeulike20&path=ASIN/0521833787>.
- Boyd, Stephen and Lieven Vandenberghe (2009). *Convex Optimization*. Seventh. Cambridge University Press.
- Braun, William (Jan. 2002). "The System Archetypes The System Archetypes". In: *System* 2002.
- Bray, Tim, Jean Paoli, and C. M. Sperberg-McQueen (Nov. 1997). "Extensible Markup Language". In: *World Wide Web J.* 2.4, pp. 29–66. ISSN: 1085-2301. URL: <http://dl.acm.org/citation.cfm?id=274784.273625>.
- Brendel, E. and C. Jäger (2005). *Contextualisms in Epistemology*. Springer Netherlands. ISBN: 9781402038358. URL: <https://books.google.hr/books?id=aN93kzwRHWIC>.
- Brendel, Elke (2013). "Wissen". In: *Lexikon Philosophy—Hundert Grundbegriffe*. Ed. by Stefan Jordan and Christian Nimtz. Reclam, pp. 308–311.
- Brette, Romain et al. (2007). "Simulation of networks of spiking neurons: A review of tools and strategies". In: *Journal of Computational Neuroscience* 23.3, pp. 349–398. ISSN: 1573-6873. DOI: <https://doi.org/10.1007/s10827007-0038-6>. URL: <https://doi.org/10.1007/s10827-007-0038-6>.
- Bretzke, W.R. (1980). *Der Problembezug von Entscheidungsmodellen*. Einheit der Gesellschaftswissenschaften: Studien in den Grenzbereichen der Wirtschafts- und Sozialwissenschaften. Mohr. ISBN: 9783169428526. URL: https://books.google.de/books?id=ICYCo_ca-1gc.
- Breuel, T. M. (2015). *OCRopus: Python-based tools for document analysis and OCR*. URL: <https://github.com/tmbdev/ocropy>.
- Briand, Lionel et al. (July 1998). "Q-MOPP: Qualitative Evaluation of Maintenance Organizations, Processes and Products". In: *Journal of Software Maintenance* 10.4, pp. 249–278. ISSN: 1040-550X. DOI: [https://doi.org/10.1002/\(SICI\)1096-908X\(199807/08\)10:4<249::AID-SMR172>3.3.CO;2-Z](https://doi.org/10.1002/(SICI)1096-908X(199807/08)10:4<249::AID-SMR172>3.3.CO;2-Z). URL: [http://dx.doi.org/10.1002/\(SICI\)1096-908X\(199807/08\)10:4<249::AID-SMR172>3.3.CO;2-Z](http://dx.doi.org/10.1002/(SICI)1096-908X(199807/08)10:4<249::AID-SMR172>3.3.CO;2-Z).
- Brocke, J. vom (2002). "Referenzmodellierung. Gestaltung und Verteilung von Konstruktionsprozesse". PhD thesis. Logos Verlag, Berlin: Univ. Münster.
- Brocke, Jan vom (2003). "Verteilte Referenzmodellierung (VRM)—Gestaltung multipersoneller Konstruktionsprozesse". In: *INFORMATIK 2003—Innovative Informatikanwendungen*

- gen, Band 1, Beiträge der 33. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 29. September—2. Oktober 2003 in Frankfurt am Main, pp. 238–242. URL: <http://subs.emis.de/LNI/Proceedings/Proceedings34/article923.html>.
- Broomhead, D. and D. Lowe (1988). “Multivariate functional interpolation and adaptive networks”. In: *Complex Systems* 2, pp. 321–355.
- Bucci, G. et al. (1994). “An object-oriented dual language for specifying reactive systems”. In: *Proceedings of the First International Conference on Requirements Engineering*, pp. 6–15. DOI: <https://doi.org/10.1109/ICRE.1994.292409>.
- Budgen, David and Pearl Brereton (2006). “Performing Systematic Literature Reviews in Software Engineering”. In: *Proceedings of the 28th International Conference on Software Engineering*. ICSE '06. Shanghai, China: ACM, pp. 1051–1052. ISBN: 1-59593-375-1. DOI: <https://doi.org/10.1145/1134285.1134500>. URL: <http://doi.acm.org/10.1145/1134285.1134500>.
- Bullinger, H.-J. and P. Schreiner (2008). *Business Process Management Tools—Eine evaluierende Marktstudie über aktuelle Werkzeuge*. IAO—Fraunhofer Institut für Arbeitswirtschaft und Organisation.
- Buresch, M., M. Kirmair, and A. Cerny (1997). “Auswahl von Organisations-Engineering-Tools”. In: *zfo* 66.1, pp. 367–373.
- Bussler, C. (2013). *B2B Integration: Concepts and Architecture*. Springer Berlin Heidelberg. ISBN: 9783662051696. URL: <https://books.google.de/books?id=r-qpCAAAQBAJ>.
- Byrd, R. H. et al. (1995). “A limited memory algorithm for bound constrained optimization”. In: *SIAM Journal on Scientific Computing* 16.6, pp. 1190–1208. URL: cseer.ist.psu.edu/byrd94limited.html.
- C4ISR (1997). *C4ISR Architecture Framework, Version 2.0*. Tech. rep. Arlington County, VA: Defense Information Systems Agency.
- Cannon, Robert C. et al. (2014). “LEMS: a language for expressing complex biological models in concise and hierarchical form and its use in underpinning NeuroML 2”. In: *Frontiers in Neuroinformatics* 8, p. 79. ISSN: 1662-5196. DOI: <https://doi.org/10.3389/fninf.2014.00079>. URL: <https://www.frontiersin.org/article/10.3389/fninf.2014.00079>.
- Cannon, Robert C. et al. (Nov. 2018). *LEMS Repository*. URL: <https://github.com/LEMS/LEMS> (visited on 08/26/2019).
- Carnevale, Nicholas T. and Michael L. Hines (2006). *The NEURON Book*. New York, NY, USA: Cambridge University Press. ISBN: 0521843219.
- Casalino, G. et al. (2016). “ANN modelling to optimize manufacturing processes: the case of laser welding”. In: *IFAC-PapersOnLine* 49.12. 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016, pp. 378–383. ISSN: 2405-8963. URL: <http://www.sciencedirect.com/science/article/pii/S2405896316309077>.
- Case, Albert F. (Sept. 1985). “Computer-aided Software Engineering (CASE): Technology for Improving Software Development Productivity”. In: *SIGMIS Database* 17.1, pp. 35–43. ISSN: 0095-0033. DOI: <https://doi.org/10.1145/1040694.1040698>. URL: <http://doi.acm.org/10.1145/1040694.1040698>.
- Cattell, R.B. (1943). *The Description of Personality: Basic Traits Resolved Into Clusters*. American psychological Association. URL: <https://books.google.de/books?id=o2zUSAAACAAJ>.
- Cawley, Gavin C and Nicola LC Talbot (2003). “Efficient leave-one-out cross-validation of kernel fisher discriminant classifiers”. In: *Pattern Recognition* 36.11, pp. 2585–2592.

- Chambers, M. and C.A. Mount-Campbell (2002a). "Process Optimization via Neural Network Metamodeling". In: *Int. J. Production Economics* 79, pp. 93–100.
- Chambers, M. and C.A. Mount-Campbell (2002b). "Process optimization via neural network metamodeling". In: *International Journal of Production Economics* 79.2. Theoretical Approaches and Decision Support, pp. 93–100. ISSN: 0925-5273. DOI: [https://doi.org/10.1016/S0925-5273\(00\)00188-2](https://doi.org/10.1016/S0925-5273(00)00188-2). URL: <http://www.sciencedirect.com/science/article/pii/S0925527300001882>.
- Chan, Chung et al. (2016). "Info-Clustering: A Mathematical Theory for Data Clustering". In: *CoRR* 1605. 01233. arXiv: 1605.01233. URL: <http://arxiv.org/abs/1605.01233>.
- Chen, CR, HS Ramaswamy, and I Alli (2001). "Prediction of quality changes during osmoconvective drying of blueberries using neural network models for process optimization". In: *Drying Technology* 19.3–4, pp. 507–523.
- Chen, Y.C. (2001). *Empirical Modelling for Participative Business Process Reengineering*. Ed. by University of Warwick (GB) and University of Warwick. Department of Computer Science. University of Warwick. URL: <https://books.google.de/books?id=M3L9MgEACA AJ>.
- Cheong, Ricky KF and Eric Tsui (2011). "From skills and competencies to outcome-based collaborative work: Tracking a decade's development of personal knowledge management (PKM) models". In: *Knowledge and Process Management* 18.3, pp. 175–193.
- Chesbrough, H., W. Vanhaverbeke, and J. West (2006). *Open Innovation: Researching a New Paradigm*. OUP Oxford. ISBN: 9780199290727. URL: <https://books.google.de/books?id=74a3AAAAIAAJ>.
- Chinosi, Michele and Alberto Trombetta (Jan. 2012). "BPMN". In: *Comput. Stand. Interfaces* 34.1, pp. 124–134. ISSN: 0920-5489. DOI: <https://doi.org/10.1016/j.csi.2011.06.002>. URL: <https://doi.org/10.1016/j.csi.2011.06.002>.
- Chiola, G. et al. (Feb. 1993). "Generalized Stochastic Petri Nets: A Definition at the Net Level and Its Implications". In: *IEEE Trans. Softw. Eng.* 19.2, pp. 89–107. ISSN: 0098-5589. DOI: <https://doi.org/10.1109/32.214828>. URL: <https://doi.org/10.1109/32.214828>.
- Chrobok, R. and E. Tiemeyer (1996). "Geschäftsprozessorganisation". In: *zfo* 65.3, pp. 165–172.
- Clark, Ruth Colvin et al. (2006). "Efficiency in learning: Evidence-based guidelines to manage cognitive load". In: *Performance Improvement* 45.9, pp. 46–47. DOI: <https://doi.org/10.1002/pfi.4930450920>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/pfi.4930450920>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/pfi.4930450920>.
- Clauberg, Katharina and William Thomas (2017). *BPM and Simulation*. A White Paper 2nd Edition. Signavio Inc.
- Cohen, D. (1972). "Magnetoencephalography: Detection of brain's electric activity with a superconducting magnetometer". In: *Science* 175, pp. 664–666.
- Cohen, Wesley M. and Daniel A. Levinthal (1990). "Absorptive Capacity: A New Perspective on Learning and Innovation". In: *Administrative Science Quarterly* 35.1, pp. 128–152.
- Colburn, T. and G Shute (2010). "Knowledge, truth, and values in computer science". In: *Thinking Machines and the Philosophy of Computer Science: Concepts and Principles*. Ed. by J. Vallverdu. IGI Global Publishing, pp. 119–131.
- Colburn, T. R. (2000). "Information, thought, and knowledge". In: *Proceedings of the world multicconference on systemics, cybernetics and informatics*, pp. 467–471.

- Costa, Daniele and Alain Hertz (1997). "Ants can colour graphs". In: *Journal of the operational research society* 48.3, pp. 295–305.
- Cox, Jonathan (2016). *CortexSys: Matlab and Octave GPU Accelerated Deep Learning Toolbox*. URL: <https://github.com/joncox123/Cortexsys>.
- Craig, E. and W. Vossenkühl (1993). *Was wir wissen Können: pragmatische Untersuchungen zum Wissensbegriff : Wittgenstein-Vorlesungen der Universität Bayreuth*. Suhrkamp-Taschenbuch Wissenschaft / Suhrkamp-Taschenbuch Wissenschaft. Suhrkamp Verlag KG. ISBN:9783518286906. URL: https://books.google.de/books?id=_yfXAAAAMAAJ.
- Creative Commons Attribution License. *CC-BY 4.0*.
- Creative Commons Attribution License *CC BY-NC-ND 4.0*.
- Cresswell, J.W. (2014). "Die Entwicklung der Mixed-Methods-Forschung". In: *Mixed Methods: Methodologie, Forschungsdesigns und Analyseverfahren*. Ed. by U. Kuckartz. Springer Fachmedien Wiesbaden, pp. 13–26. ISBN: 9783531932675. URL: <https://books.google.de/books?id=iQIYBAAQBAJ>.
- Crook, Sharon et al. (2007). "MorphML: Level 1 of the NeuroML Standards for Neuronal Morphology Data and Model Specification". In: *Neuroinformatics* 5.2, pp. 96–104. ISSN: 1559-0089. DOI: <https://doi.org/10.1007/s12021-007-0003-6>. URL: <https://doi.org/10.1007/s12021-007-0003-6>.
- Cummings, Jeffrey L. and Bing-Sheng Teng (2003). "Transferring R&D knowledge: the key factors affecting knowledge transfer success". In: *Journal of Engineering and Technology Management* 20.1, pp. 39–68.
- Curatelli, F. and O. Mayora-Ibarra (2000). "Competitive Learning Methods for Efficient Vector Quantizations in a Speech Recognition Environment". In: *MICAI 2000: Advances in Artificial Intelligence*. Ed. by Osvaldo Cairó, L. Enrique Sucar, and Francisco J. Cantu. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 108–114. ISBN: 978-3-540-45562-2.
- Curtis, Bill, Marc I. Kellner, and Jim Over (Sept. 1992). "Process Modeling". In: *Commun. ACM* 35.9, pp. 75–90. ISSN: 0001-0782. DOI: <https://doi.org/10.1145/130994.130998>. URL: <http://doi.acm.org/10.1145/130994.130998>.
- Cyert, R.M. and J.G. March (1963). *A behavioral theory of the firm*. Prentice-Hall behavioral sciences in business series. Prentice-Hall. URL: <https://books.google.de/books?id=5YxEAAAIAAJ>.
- Daft, Richard and George Huber (1986). "How organizations learn: a communication framework." In: *Research in the Sociology of Organizations* 5, pp. 1–36.
- Daft, Richard L. and Robert H. Lengel (1984). "Information richness: a new approach to managerial behavior and organization design". In: *Research in Organizational Behavior* 6, pp. 191–233.
- Dagkakis, Georgios et al. (2013). "ManPy: An open-source layer of DES manufacturing objects implemented in SimPy". In: *2013 8th EUROSIM Congress on Modelling and Simulation*. IEEE, pp. 357–363.
- Dahl, O. J., E. W. Dijkstra, and C. A. R. Hoare, eds. (1972). *Structured Programming*. London, UK: Academic Press Ltd. ISBN: 0-12-200550-3.
- Dämming, I., U. Hess, and C. Borgmann (2002). "Geschäftsprozessorientiertes Wissensmanagement: effektive Wissensnutzung bei der Planung und Umsetzung von Geschäftssprozessen". In: ed. by A. Ab-decker et al. Xpert.press. Berlin Heidelberg: Springer Berlin Heidelberg. Chap. Kommunikationsdiagnose (KODA)—Einstiegsmethode und -werkzeug in das praktische Wissensmanagement, pp. 123–158.

- Davenport, T.H. (1993). *Process Innovation: Reengineering Work Through Information Technology*. Harvard Business Review Press. ISBN: 9781422160664. URL: <https://books.google.de/books?id=kLlIOMGaKnsC>.
- Davenport, T.H. (1994). "Saving IT's Soul: Human-Centered Information Management". In: *Harvard Business Review* 72.2, pp. 119–131.
- Davenport, T.H., S.L. Jarvenpaa, and W.C.M.V. Beers (1996). "Improving Knowledge Work Processes". In: *Sloan Management Review* 37 4, pp. 53–65.
- Davenport, T.H. and L. Prusak (2000). *Working Knowledge: How Organizations Manage what They Know*. Knowledge Management. Harvard Business School Press. ISBN: 9781578513017. URL: <https://books.google.de/books?id=QIyIWVhdYoYC>.
- Davenport, Thomas H. and James E. Short (1990). "The New Industrial Engineering: Information Technology and Business Process Redesign". In: *Sloan Management Review* 31.4, pp. 11–27. URL: <http://sloanreview.mit.edu/smr/issue/1990/summer/1/>.
- Davis, Gerald Wesley and Michael L Gasperi (1997). *Process optimization using a neural network*. US Patent 5,671,335.
- Davis, R. (2012). *Business Process Modelling with ARIS: A Practical Guide*. Springer London. URL: <https://books.google.de/books?id=Bm4QBwAAQBAJ>.
- Davison, Andrew et al. (Nov. 2005). *NeuroML Repository*. URL: <https://sourceforge.net/projects/neuroml/> (visited on 08/14/2019).
- De Schutter, E. and J. M. Bower (1994). "An active membrane model of the cerebellar Purkinje cell. I. Simulation of current clamps in slice". In: *Journal of Neurophysiology* 71.1. PMID: 7512629, pp. 375–400. DOI: <https://doi.org/10.1152/jn.1994.71.1.375>. eprint: <https://doi.org/10.1152/jn.1994.71.1.375>. URL: <https://doi.org/10.1152/jn.1994.71.1.375>.
- Dean, Jeffrey and Sanjay Ghemawat (Jan. 2008). "MapReduce: Simplified Data Processing on Large Clusters". In: *Commun. ACM* 51.1, pp. 107–113. ISSN: 0001-0782. DOI: <https://doi.org/10.1145/1327452.1327492>. URL: <http://doi.acm.org/10.1145/1327452.1327492>.
- Decker, Gero, Hagen Overdick, and Mathias Weske (2008). "Oryx—An Open Modeling Platform for the BPM Community". In: *Business Process Management*. Ed. by Marlon Dumas, Manfred Reichert, and Ming-Chien Shan. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 382–385. ISBN: 978-3-540-85758-7.
- Defense, United States. Department of (1949). *Procedures for Performing a Failure Mode, Effects and Criticality Analysis*. Tech. rep. MIL-P-1629. Department of Defense (US).
- Defense, United States. Department of (1980). *Procedures for Performing a Failure Mode, Effects and Criticality Analysis*. MIL STD 1629A : 1980. U.S. Department of Defense. URL: <https://books.google.de/books?id=ncajSwAACAAJ>.
- Deming, W. Edwards (1993). *The new economics for industry, government, education / W. Edwards Deming*. English. Massachusetts Institute of Technology, Center for Advanced Engineering Study Cambridge, MA, xii, 240 p. : ISBN: 0911379053.
- Deming, W.E. (1950). *Elementary Principles of the Statistical Control of Quality: A Series of Lectures*. Nippon Kagaku Gijutsu Remmei. URL: <https://books.google.de/books?id=lZhXnQAACAAJ>.
- Deming, W.E. (1986). *Out of the Crisis*. Cambridge University Press. ISBN: 9780521305532. URL: <https://books.google.de/books?id=4qw8AAAAIAAJ>.
- Demuth, Howard and Mark Beale (1993). *Neural Network Toolbox For Use with Matlab—User'S Guide Verion 3.0*.

- DePaul, Michael and Linda Zagzebski (2003). *Intellectual Virtue: Perspectives From Ethics and Epistemology*. Oxford University Press.
- Deru, M. and A. Ndiaye (2019). *Deep Learning mit TensorFlow, Keras und TensorFlow.js: Einstieg, Konzepte und KI-Projekte mit Python, JavaScript und HTML5*. Rheinwerk Computing. Rheinwerk Verlag GmbH. ISBN: 9783836265096. URL: <https://books.google.de/books?id=NFihvwEACAAJ>.
- Desel, Jörg and Wolfgang Reisig (1996). "Place or Transition Petri Nets". In: *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets, held in Dagstuhl, September 1996*, pp. 122–173. DOI: https://doi.org/10.1007/3-540-65306-6_15. URL: https://doi.org/10.1007/3-540-65306-6_15.
- Desel, Jörg and Wolfgang Reisig (1998). "Place/transition Petri Nets". In: *Lectures on Petri Nets I: Basic Models: Advances in Petri Nets*. Ed. by Wolfgang Reisig and Grzegorz Rozenberg. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 122–173. ISBN: 978-3-540-49442-3. DOI: https://doi.org/10.1007/3-540-653066_15. URL: https://doi.org/10.1007/3-540-653066_15.
- Desouza, Kevin C. and Yukika Awazu (2006). "Knowledge management at SMEs: five peculiarities". In: *Journal of Knowledge Management* 10.1, pp. 32–43. DOI: <https://doi.org/10.1108/13673270610650085>. eprint: <https://doi.org/10.1108/13673270610650085>. URL: <https://doi.org/10.1108/13673270610650085>.
- Desouza, Kevin C et al. (2005). "Facilitating knowledge management through market mechanism". In: *Knowledge and Process Management* 12.2, pp. 99–107.
- Deuse, J., C. Goldscheid, and Y. Finke (2007). "Prozesskostenrechnung in logistischen Bereichen". In: *Industrie Management* 5.
- Diamadopoulos, Peter (1960). "Reviewed Work: The Presocratic Philosophers. A Critical History with a Selection of Texts by G. S. Kirk, J. E. Raven". In: *The American Journal of Philology* 81.1, pp. 100–104. ISSN: 00029475, 10863168. URL: <http://www.jstor.org/stable/291768>.
- Dibella, Anthony J. (1995). "Sculpting the learning organization: Lessons in the art and science of systemic change, by Karen E. Watkins and Victoria J. Marsick. (1993). San Francisco: Jossey-Bass. 298 pp." In: *Human Resource Development Quarterly* 6.4, pp. 421–423. DOI: <https://doi.org/10.1002/hrdq.3920060410>.
- Diekmann, A. (2012). *Empirische Sozialforschung: Grundlagen, Methoden, Anwendungen*. Vol. 6. Hamburg: Rowohlt-Taschenbuch-Verlag.
- Diels, Hermann and Walther Kranz (1906). *Die Fragmente der Vorsokratiker Griechisch Und Deutsch*. Weidmann.
- Dierkes, Meinolf et al. (2001). *Handbook of Organizational Learning and Knowledge*. USA: Oxford University Press, Inc. ISBN: 0198295839.
- Diesmann, Markus and Marc oliver Gewaltig (2002). "NEST: An environment for neural systems simulations". In: *In T. Plessner and V. Macho (Eds.), Forschung und wissenschaftliches Rechnen, Beiträge zum Heinz-Billing-Preis 2001, Volume 58 of GWDG-Bericht*, pp. 43–70.
- Diestel, R. (1996). *Graphentheorie*. Springer-Lehrbuch. Springer. ISBN: 9783540609186. URL: <https://books.google.de/books?id=ivjuAAAAMAAJ>.
- Disterer, Georg (2000). "Individuelle und soziale Barrieren beim Aufbau von Wissenssammlungen". In: *Wirtschaftsinformatik* 42.6, pp. 539–546. ISSN: 1861-8936. DOI: <https://doi.org/10.1007/BF03250771>. URL: <https://doi.org/10.1007/BF03250771>.

- Dixon, Nancy M. (1992). "Organizational learning: A review of the literature with implications for HRD professionals". In: *Human Resource Development Quarterly* 3.1, pp. 29–49.
- DoDAF (2007a). *The DoDAF Architecture Framework, Version 1.5*. Tech. rep. (Volume I: Definitions and Guidelines). Arlington County, VA: Defense Information Systems Agency.
- DoDAF (2007b). *The DoDAF Architecture Framework, Version 1.5*. Tech. rep. (Volume II: Product Descriptions). Arlington County, VA: Defense Information Systems Agency.
- DoDAF (2007c). *The DoDAF Architecture Framework, Version 1.5*. Tech. rep. (Volume III: Architecture Data Description). Arlington County, VA: Defense Information Systems Agency.
- DoDAF (2007d). *The DoDAF Architecture Framework, Version 2.0*. Tech. rep. Arlington County, VA: Defense Information Systems Agency.
- Domínguez, E., M. A. Zapata, and J. Rubio (1997). "A conceptual approach to meta-modelling". In: *Advanced Information Systems Engineering*. Ed. by Antoni Olivé and Joan Antoni Pastor. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 319–332. ISBN: 978-3-540-69148-8.
- Dor, Ofer and Yaoqi Zhou (2007). "Achieving 80% ten-fold-validated accuracy for secondary structure prediction by large-scale training". In: *Proteins: Structure, Function, and Bioinformatics* 66.4, pp. 838–845.
- Doran, Jim (1997). "From computer simulation to artificial societies". In: *Transactions of the Society for Computer Simulation International* 14.2, pp. 69–77.
- Dorigo, Marco, Gianni Di Caro, and Luca M. Gambardella (Apr. 1999). "Ant Algorithms for Discrete Optimization". In: *Artif. Life* 5.2, pp. 137–172. ISSN: 1064-5462. DOI: <https://doi.org/10.1162/106454699568728>. URL: <http://dx.doi.org/10.1162/106454699568728>.
- Dorigo, Marco and Thomas Stützle (2003). "The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances". In: *Handbook of Metaheuristics*. Ed. by Fred Glover and Gary A. Kochenberger. Boston, MA: Springer US, pp. 250–285. ISBN: 978-0-306-48056-0. DOI: https://doi.org/10.1007/0-306-48056-5_9. URL: https://doi.org/10.1007/0-306-48056-5_9.
- Dörner, Dietrich (1999). "Sozialwissenschaftliche Methoden: Lehr- und Handbuch für Forschung und Praxis". In: ed. by E. Roth, H. Holling, and K. Heidenreich. Lehr- und Handbücher der Sozialwissenschaften. Oldenbourg: Oldenbourg. Chap. Modellbildung und Simulaiton, pp. 327–340. ISBN: 9783486252637. URL: <https://books.google.de/books?id=ax-gAAAACAAJ>.
- Drawehn, J., S. Gayer, and P. Schneider (2010). *Business Process Modeling—Modellierung von ausführbaren Geschäftsprozessen mit der Business Process Modeling Notation*. Fraunhofer Verlag.
- Drawehn, J. et al. (2014). *Business Process Management Tools 2014: Marktüberblick*. Business Process Management Tools 2014. Fraunhofer Verlag. ISBN: 9783839607763. URL: <https://books.google.de/books?id=sY3GoQEACAAJ>.
- Driver, H.E. and A.L. Kroeber (1932). *Quantitative Expression of Cultural Relationships*. Publications in American archaeology and ethnology. University of California Press. URL: <https://books.google.de/books?id=I9SjDQEACAAJ>.
- Dueck, G. (1989). *New Optimization Heuristics: The Great Deluge Algorithm and the Record-to-record Travel*. Technical reports. IBM Deutschland GmbH. URL: <https://books.google.de/books?id=ezDcPgAACAAJ>.

- Dueck, G. (2004). *Das Sintflutprinzip: Ein Mathematik-Roman*. Springer Berlin Heidelberg.
ISBN: 9783662064917. URL: <https://books.google.no/books?id=4gj9jwEACAAJ>.
- Dueck, Gunter (1993). "New optimization heuristics: The great deluge algorithm and the record-to-record travel". In: *Journal of Computational physics* 104.1, pp. 86–92.
- Dueck, Gunter and Tobias Scheuer (1990). "Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing". In: *Journal of computational physics* 90.1, pp. 161–175.
- Dumas, M. et al. (2013). *Fundamentals of Business Process Management*. Springer, Heidelberg.
- Dumas, Marlon, Wil M. van der Aalst, and Arthur H. ter Hofstede (2005). *Process-aware Information Systems: Bridging People and Software Through Process Technology*. New York, NY, USA: John Wiley & Sons, Inc. ISBN: 0-47166-360-9.
- Duncan, Robert and A. Weiss (1979). "Organizational learning: Implications for organizational design". In: *Research in organizational behavior* 1, pp. 75–123.
- Dunie, Rob et al. (2015). *Magic Quadrant for Intelligent Business Process Management Suite*. Tech. rep. G00258612. Gartner Inc.
- Dunie, Rob et al. (2016). *Magic Quadrant for Intelligent Business Process Management Suite*. Tech. rep. G00276892. Gartner Inc.
- Dunie, Rob et al. (2017). *Magic Quadrant for Intelligent Business Process Management Suite*. Tech. rep. G00315642. Gartner Inc.
- Eberhard, J.A. (1802). *Johann August Eberhards synonymisches Handwörterbuch der deutschen Sprache: für alle, die sich in dieser Sprache richtig ausdrücken wollen ; nebst einer ausführlichen Anweisung zum nützlichen Gebrauche desselben*. Schimmelpfennig.
URL: <https://books.google.de/books?id=vBBJAAAACAAJ>.
- Eberhart, Russell and James Kennedy (1995). "A new optimizer using particle swarm theory". In: *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. Ieee, pp. 39–43.
- Eck, D. and J. Schmidhuber (2002a). "A First Look at Music Composition using LSTM Recurrent Neural Networks". In: *Technical Report No. IDSIA-07-02*, p. 1.
- Eck, D. and J Schmidhuber (2002b). "Learning The Long-Term Structure of the Blues". In: *Artificial Neural Networks—ICANN 2002*, pp. 284–289.
- Eherer, S. (1995). *Eine Software-Umgebung für die kooperative Erstellung von Hypertexten. Sprache und Information*. Niemeyer. ISBN: 9783484319295. URL: <https://books.google.de/books?id=auSmPQAACAAJ>.
- Eisenhardt, Kathleen M and Filipe M Santos (2002). "Knowledge-based view: A new theory of strategy". In: *Handbook of strategy and management* 1.1, pp. 139–164.
- Ellis, A. and M. Kauferstein (2012). *Dienstleistungsmanagement: Erfolgreicher Einsatz von prozessorientiertem Service Level Management*. Springer Berlin Heidelberg. ISBN: 9783642170072. URL: <https://books.google.de/books?id=UX8jBgAAQBAJ>.
- Engeln, W. and J. Schwöbel (2005). *Prozesse optimieren mit Wertstromdesign*. Gito Verlag Berlin.
- Eppler, M. et al. (2000). *Das Enterprise Knowledge Media Referenzmodell*. HSG/MCM/Nr. 6—Work Report 40. Universität St. Gallen. Kompetenzzentrum EKM.
- Eppler, Martin J (1997). "Führer durch den Wissensschubel". In: *Gablers Magazin* 8, p. 97.
- Eppler, Martin J. (2004). "Making Knowledge Visible through Knowledge Maps: Concepts, Elements, Cases". In: *Handbook on Knowledge Management I: Knowledge Matters*. Ed.

- by Clyde W. Holsapple. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 189–205. ISBN: 978-3-540-24746-3. DOI: https://doi.org/10.1007/978-3-54024746-3_10. URL: https://doi.org/10.1007/978-3-540-24746-3_10.
- Ergazakis, Kostas, Kostas Metaxiotis, and Dimitris Askounis (2013). “Knowledge-based development research: a comprehensive literature review 2000–2010”. In: *Knowledge management research & practice* 11.1, pp. 78–91.
- Erhan, Dumitru et al. (Jan. 2009). “Visualizing Higher-Layer Features of a Deep Network”. In: *Technical Report, University of Montreal*.
- Ernst, G. (2002). Das Problem des Wissens. Mentis. ISBN: 9783897852624. URL: <https://books.google.de/books?id=bHl-swEACAAJ>.
- Esparza, J. (Oct. 1994). “Reduction and Synthesis of Live and Bounded Free Choice Petri Nets”. In: *Inf. Comput.* 114.1, pp. 50–87. ISSN: 0890-5401. DOI: <https://doi.org/10.1006/inco.1994.1080>. URL: <http://dx.doi.org/10.1006/inco.1994.1080>.
- Esparza, Javier (1998). “Decidability and complexity of Petri net problems—An introduction”. In: *Lectures on Petri Nets I: Basic Models: Advances in Petri Nets*. Ed. by Wolfgang Reisig and Grzegorz Rozenberg. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 374–428. ISBN: 978-3-540-49442-3. DOI: https://doi.org/10.1007/3540-65306-6_20. URL: https://doi.org/10.1007/3-540-65306-6_20.
- Esparza, Javier and Keijo Heljanko (2008). *Unfoldings—A Partial-Order Approach to Model Checking*. Monographs in Theoretical Computer Science. An EATCS Series. Springer. ISBN: 978-3-540-77425-9. DOI: <https://doi.org/10.1007/978-3-540-77426-6>. URL: <https://doi.org/10.1007/978-3-540-77426-6>.
- Ev3Org (Sept. 2020a). *ev3dev is your EV3 re-imagined*. URL: <https://www.ev3dev.org/>.
- Ev3Org (Sept. 2020b). *ev3dev is your EV3 re-imagined*. URL: <https://www.ev3dev.org/news/2019/04/13/ev3-micropython/>.
- Fahlman, S. (1989). “Faster learning variations on back-propagation: An empirical study”. In: *Proceedings of the 1988 connectionist models summer school*, In D. Touretzky, G. Hinton and T. Sejnowski, editors, San Mateo, Morgan Kaufmann, pp. 38–51.
- Fahlman, S. E. (1991). “The recurrent cascade-correlation learning algorithm”. In: *Advances in Neural Information Processing Systems* 3, pp. 190–196. URL: <http://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735?journalCode=neco>.
- Fanchao, Zeng, Stephen John Turner, and Heiko Aydt (2009). “Symbiotic Simulation Control in Supply Chain of Lubricant Additive Industry”. In: *13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications, Singapore, 25–28 October 2009*, pp. 165–172. DOI: <https://doi.org/10.1109/DS-RT.2009.17>. URL: <https://doi.org/10.1109/DS-RT.2009.17>.
- Fank, M. (2013). *Tools zur Geschäftsprozeßorganisation: Entscheidungskriterien, Fallstudienorientierung, Produktvergleiche*. XBusiness Computing. Vieweg+Teubner Verlag. ISBN: 9783322906670. URL: <https://books.google.de/books?id=mFHNBgAAQBAJ>.
- FEA (1999). *FEA: A Practical Guide to Federal Enterprise Architecture, Version 1.0*. Tech. rep. Springfield, VA: Chief Information Officer Council.
- FEAF (1999). *FEAF: Federal Enterprise Architecture Framework, Version 1.1*. Tech. rep. Springfield, VA: Chief Information Officer Council.
- Feilmayr, Christina and Wolfram Wös (2016). “An analysis of ontologies and their success factors for application to business”. In: *Data & Knowledge Engineering* 101, pp. 1–23.
- Feldman, Richard (1985). “Reliability and Justification”. In: *The Monist* 68.2, pp. 159–174.

- Fernández, Joaquín Lopez et al. (2008). “Using hierarchical binary Petri nets to build robust mobile robot applications: RoboGraph.” In: *ICRA*. IEEE, pp. 1372–1377. URL: <http://dblp.uni-trier.de/db/conf/icra/icra2008.html#FernandezSDA08>
- Ferstl, O.K. and E.J. Sinz (1990). “Objektmodellierung betrieblicher Informationssysteme im Semantischen Objektmodell”. In: *Wirtschaftsinformatik* 6, pp. 566–581.
- Ferstl, O.K. and E.J. Sinz (1993). “Geschäftsprozeßmodellierung”. In: *Wirtschaftsinformatik* 35.
- Ferstl, O.K. and E.J. Sinz (2001). *Grundlagen der Wirtschaftsinformatik*. 4 1. Oldenbourg Verlag München.
- Fettke, P. (2007). “Referenzmodellevaluation—Konzeption der strukturalistischen Referenzmodellierung und Entfaltung ontologischer Gütekriterien”. In: *WIRTSCHAFTSINFORMATIK* 49.3, pp. 224–224. ISSN: 1861-8936. DOI: <https://doi.org/10.1007/s11576-007-0105-7>. URL: <https://doi.org/10.1007/s11576007-0105-7>.
- Fettke, P. and J. vom Brocke (2016a). “Referenzmodell”. In: *Enzyklopädie der Wirtschaftsinformatik—Online-Lexikon*. Ed. by K. Kurbel et al. Vol. 8. Gito Verlag Berlin. URL: <http://www.enzyklopaedie-derwirtschaftsinformatik.de/lexikon/is-management/Systementwicklung/Softwarearchitektur/Wiederverwendung-von-Softwarebausteinen/Referenzmodell/index.html?searchterm=referenzmodell>.
- Fettke, P. and J. vom Brocke (2016b). “Referenzmodellierung”. In: *Enzyklopädie der Wirtschaftsinformatik—Online-Lexikon*. Ed. by K. Kurbel et al. Vol. 8. Gito Verlag Berlin. URL: <http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/datenwissen/Informationsmanagement/referenzmodellierung>.
- Fettke, Peter (2020). “Conceptual Modelling and Artificial Intelligence: Overview and research challenges from the perspective of predictive business process management”. In: *Companion Proceedings of Modellierung 2020 Short, Workshop and Tools & Demo Papers co-located with Modellierung 2020, Vienna, Austria, February 19-21, 2020*, pp. 157–164. URL: <http://ceur-ws.org/Vol-2542/MOD-KI4.pdf>.
- Fettke, Peter, Houy Constantin, and Loos Peter (2010a). *Zur Bedeutung von Gestaltungswissen für die gestaltungsorientierte Wirtschaftsinformatik—Ergänzende Überlegungen und weitere Anwendungsbeispiele*. Arbeitsbericht 191, Institut für Wirtschaftsinformatik im Deutschen Forschungszentrum für Künstliche Intelligenz (DFKI), Saarbrücken.
- Fettke, Peter, Houy Constantin, and Loos Peter (2010b). *Zur Bedeutung von Gestaltungswissen für die gestaltungsorientierte Wirtschaftsinformatik. Konzeptionelle Grundlagen, Anwendungsbeispiel und Implikationen*. Wirtschaftsinformatik, 52(6): 339–352, 2010.
- Fettke, Peter and Peter Loos (2004a). “Entwicklung eines Bezugsrahmens zur Evaluierung von Referenzmodellen—Langfassung eines Beitrags”. In: *ISYM—Information Systems and Management* 20.
- Fettke, Peter and Peter Loos (2004b). “Referenzmodellierungsorschung”. In: *WIRTSCHAFTSINFORMATIK* 46.5, pp. 331–340. ISSN: 1861-8936. DOI: <https://doi.org/10.1007/BF03250947>. URL: <https://doi.org/10.1007/BF03250947>.
- Fettke, Peter and Peter Loos (2007). “Reference Modeling for Business Systems Analysis”. In: ed. by Peter Fettke and Peter Loos. IGI Globa. Chap. Perspectives on Reference Modeling, pp. 1–21. DOI: <https://doi.org/10.4018/978-1-59904-0547.ch001>.
- Finkeißen, A., M. Forschner, and M. Häge (1996). “Werkzeuge zur Prozessanalyse und -optimierung”. In: *Controlling* 8.1, pp. 58–67.

- Fischer, Christian (2010). "Auf dem Weg zu Kriterien zur Auswahl einer geeigneten Evaluationsmethode für Artefakte der gestaltungsorientierten Wirtschaftsinformatik". In: *Lecture Notes in Informatics, Conference: EMISA 2010—Einflussfaktoren auf die Entwicklung flexibler, integrierter Informationssysteme. Beiträge des Workshops der GI-Fachgruppe Entwicklungsmethoden für Informationssysteme und deren Anwendung (EMISA), 07.–08.10.2010*. Ed. by Stefan Klink et al. Vol. 172. 1. Karlsruhe, Germany: Gesellschaft für Informatik.
- Fischermanns, G. (2013). *Praxishandbuch Prozessmanagement : das Standardwerk auf Basis des BPM Framework ibo-Prozessfenster®*. Ibo Schriftenreihe. Dr. Götz Schmidt. ISBN: 9783921313893. URL: <https://books.google.de/books?id=ZffpngEACAAJ>.
- Fischler, Martin A. and Robert C. Bolles (June 1981). "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Commun. ACM* 24.6, pp. 381–395. ISSN: 0001-0782. DOI: <https://doi.org/10.1145/358669.358692>. URL: <http://doi.acm.org/10.1145/358669.358692>.
- Fishman, G. (2001). *Discrete-Event Simulation: Modeling, Programming, and Analysis*. Springer Series in Operations Research and Financial Engineering. Springer New York. ISBN: 9780387951607. URL: <https://books.google.de/books?id=PA3lj2KB3uEC>.
- Floyd, C. and R. Klischewski (1998). "Modellierung—ein Handgriff zur Wirklichkeit: Zur sozialen Konstruktion und Wirksamkeit von Informatik-Modellen". In: *Modellierung '98: Proceedings GI-Workshop, Münster, 11.–13. März 1998*. Ed. by K. Pohl, A. Schürr, and G. Vossen. Universität Münster.
- Fong, Ruth and Andrea Vedaldi (2017). "Interpretable Explanations of Black Boxes by Meaningful Perturbation". In: *CoRR* abs/1704.03296. arXiv:1704.03296. URL: <http://arxiv.org/abs/1704.03296>.
- Forrester, Jay W. (1961). *Industrial dynamics*. M.I.T. Press Cambridge, Mass, p. 464.
- Forrester, Jay W. (1972). *Grundzüge einer Systemtheorie*. Gabler.
- Frank, Ulrich (1992). "Designing Procedures within an Object-Oriented Enterprise Model". In: *Proceedings of the Third International Working Conference on Dynamic Modelling of Information Systems*. Ed. by H. Sol, pp. 365–388. URL: <http://www.wi-inf.uni-due.de/FGFrank/documents/Zeitschriftenartikel/DelftFinal.pdf>.
- Frank, Ulrich (1993). "Multiperspektivische Unternehmensmodellierung als Basis und Gegenstand integrierter CSCW-Systeme". PhD thesis.
- Frank, Ulrich (July 2014). "Multi-perspective Enterprise Modeling: Foundational Concepts, Prospects and Future Research Challenges". In: *Softw. Syst. Model.* 13.3, pp. 941–962. ISSN: 1619-1366. DOI: <https://doi.org/10.1007/s10270-012-0273-9>. URL: <http://dx.doi.org/10.1007/s10270-012-0273-9>.
- Frank, Ulrich and Stefan Klein (1992). *Three Integrated Tools for Designing and Prototyping Object-Oriented Enterprise Models*. Arbeitspapiere der GMD 689. URL: http://www.wi-inf.uni-due.de/FGFrank/documents/Arbeitsberichte_Koblenz/OO-Bericht.pdf.
- Frischmuth, Norman (2002). *Frischmuth, Norman: Anreizsysteme für den innerbetrieblichen Wissensmarkt. Organisatorische und technologische Möglichkeiten*. Marburg.
- Fritzke, Bernd (1995). "A Growing Neural Gas Network Learns Topologies". In: *Advances in Neural Information Processing Systems 7*. Ed. by G. Tesauro, D. S. Touretzky, and T. K. Leen. MIT Press, pp. 625–632. URL: <http://papers.nips.cc/paper/893-a-growing-neural-gas-network-learnstopologies.pdf>.

- Fröming, J. (2009). "Ein Konzept zur Simulation wissensintensiver Aktivitäten in Geschäftsprozessen". In: *GITO mbH Verlag*.
- Fröming, Jane, Norbert Gronau, and Eldar Sultanow (2008). "Softwareautomaten—Im Vergleich: jABC, AndroMDA und Open-ArchitectureWare". In: *IX—Magazin für professionelle Informationstechnik* 8, pp. 92–95.
- Fuchs-Wegner, G. (1972). "Systemtheorie". In: ed. by R. Kurzrock. Berlin: Colloquium Verlag. Chap. Verfahren der Analyse von Systemen.
- Furnell, Will et al. (2019). "First results from the LUCID-Timepix spacecraft payload onboard the TechDemoSat1 satellite in Low Earth Orbit". In: *Advances in Space Research* 63.5, pp. 1523–1540.
- Gaitanides, M. (1979a). "Konstruktion von Entscheidungsmodellen und ‘Fehler dritter Art’". In: *Wirtschaftswissenschaftliches Studium* 8.1, pp. 8–12.
- Gaitanides, M. (1979b). *Planungsmethodologie: Vorentscheidungen bei der Formulierung integrierter Investitionsplanungsmodelle*. Ed. by Technische Hochschule Darmstadt. Berlin: Duncker und Humblot.
- Gaitanides, M. (1994). *Prozessmanagement: Konzepte, Umsetzungen und Erfahrungen des Reengineering*. München, Wien: Hanser. ISBN: 9783446177154. URL: <https://books.google.de/books?id=jjHCAAAACAAJ>.
- GAO (1992). *Strategic Information Planning: Framework for Designing and Developing System Architectures*. Tech. rep. GAO/IMTEC-92-51. Washington, DC: Government Accountability Office.
- Garcia-Cantero, Juan J. et al. (2017). "NeuroTessMesh: A Tool for the Generation and Visualization of Neuron Meshes and Adaptive On-the-Fly Refinement". In: *Frontiers in Neuroinformatics* 11, p. 38. ISSN: 1662-5196. DOI: <https://doi.org/10.3389/fninf.2017.00038>. URL: <https://www.frontiersin.org/article/10.3389/fninf.2017.00038>.
- Gardner, Daniel et al. (2001). "Common data model for neuroscience data and data model exchange". In: *Journal of the American Medical Informatics Association* 8.1, pp. 17–33.
- Gardner, Daniel et al. (2008). "Terminology for Neuroscience Data Discovery: Multi-tree Syntax and Investigator-Derived Semantics". In: *Neuroinformatics* 6.3, pp. 161–174. URL: <https://doi.org/10.1007/s12021-008-9029-7>.
- Gare, Arran (1999). "Speculative metaphysics and the future of philosophy: The contemporary relevance of whitehead's defence of speculative metaphysics". In: *Australasian Journal of Philosophy* 77.2, pp. 127–145. DOI: <https://doi.org/10.1080/0048409912348891>. eprint: <https://doi.org/10.1080/0048409912348891>. URL: <https://doi.org/10.1080/0048409912348891>.
- Garg, Rajiv, Michael D Smith, and Rahul Telang (2011). "Measuring information diffusion in an online community". In: *Journal of Management Information Systems* 28.2, pp. 11–38.
- Gathercole, Susan E. (1998). "The Development of Memory". In: *Journal of Child Psychology and Psychiatry* 39.1, pp. 3–27. DOI: <https://doi.org/10.1111/1469-7610.00301>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/1469-7610.00301>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/1469-7610.00301>.
- Gehring, Hermann (1998). *Operation Research—Simulation—Studienheft*. Fernuniversität Hagen.
- Gemechu, Eskinder Demisse et al. (2015). "How to Implement Life Cycle Management in Business?" In: *Life Cycle Management*. Ed. by Guido Sonnemann and Manuele Margni. Dordrecht: Springer Netherlands, pp. 35–50. ISBN: 978-94-017-7221-1.

- DOI: https://doi.org/10.1007/978-94-017-7221-1_4. URL: https://doi.org/10.1007/978-94-017-7221-1_4.
- Genesis-Community (Nov. 2017). *Genesis Webpage*. URL: <http://www.genesis-sim.org/> (visited on 12/28/2018).
- Genesis-Community (Nov. 2018). *Genesis Repository*. URL: <https://github.com/genesis-sim> (visited on 12/28/2018).
- Gers F., A., J. Schmidhuber, and F. Cummins (2000). *Learning to Forget: Continual Prediction with LSTM*. Tech. rep. 10, pp. 2451–2471.
- Gettier, Edmund L. (1963). “Is Justified True Belief Knowledge?” In: *Analysis* 23.6, pp. 121–123. ISSN: 00032638, 14678284. URL: <http://www.jstor.org/stable/3326922>.
- Giaglis, George M. (2001). “A Taxonomy of Business Process Modeling and Information Systems Modeling Techniques”. In: *International Journal of Flexible Manufacturing Systems* 13.2, pp. 209–228. ISSN: 1572-9370. DOI: <https://doi.org/10.1023/A:1011139719773>. URL: <https://doi.org/10.1023/A:1011139719773>.
- Giel, Susanne (2013). *Theoriebasierte Evaluation: Konzepte und methodische Umsetzungen*. Münster: Waxmann.
- Gilbert, N. and K. Troitzsch (2005). *Simulation For The Social Scientist*. Open University Press. URL: <https://books.google.de/books?id=VflDBgAAQBAJ>.
- Gilbreth, FB and Lillian M Gilbreth (1921). “Process Charts, First Steps in Finding the One Best Way to Do Work”. In: *American Society of Mechanical Engineers, Annual Meeting, December*. Vol. 5.
- Gleeson, Padraig, Volker Steuber, and R. Angus Silver (2007). “neuroConstruct: A Tool for Modeling Networks of Neurons in 3D Space”. In: *Neuron* 54.2. Exported from <https://app.dimensions.ai> on 2018/07/24, pp. 219–235. DOI: <https://doi.org/10.1016/j.neuron.2007.03.025>. URL: <https://app.dimensions.ai/details/publication/pub.1050835493> and <https://doi.org/10.1016/j.neuron.2007.03.025>.
- Gleeson, Padraig et al. (June 2010). “NeuroML: A Language for Describing Data Driven Models of Neurons and Networks with a High Degree of Biological Detail”. In: *PLOS Computational Biology* 6.6, pp. 1–19. DOI: <https://doi.org/10.1371/journal.pcbi.1000815>. URL: <https://doi.org/10.1371/journal.pcbi.1000815>.
- GMRV, Grupo de Modelado y Realidad Virtual (engl. Virtual Reality and Modeling Group) (Nov. 2018). *NeuroLots and TessMesh Webpage*. URL: <http://gmrv.es/neurotessmesh> (visited on 11/03/2018).
- GMRV, Grupo de Modelado y Realidad Virtual (engl. Virtual Reality and Modeling Group) and Community (Nov. 2018a). *NeuroLots Repository*. URL: <https://github.com/gmrvvis/neurolots> (visited on 11/03/2018).
- GMRV, Grupo de Modelado y Realidad Virtual (engl. Virtual Reality and Modeling Group) and Community (Nov. 2018b). *TessMesh Repository*. URL: <https://github.com/gmrvvis/NeuroTessMesh> (visited on 11/03/2018).
- GNU General Public License v3.0. *GNU-GPL 3.0*.
- Goddard, Nigel H. et al. (2001). “Towards NeuroML: Model Description Methods for Collaborative Modelling in Neuroscience”. In: *Philosophical Transactions: Biological Sciences* 356.1412, pp. 1209–1228. ISSN: 09628436. URL: <http://www.jstor.org/stable/3067148>.
- Goldman, A.I. (1986). *Epistemology and Cognition*. Harvard University Press. ISBN: 9780674258969. URL: <https://books.google.de/books?id=9XwhBw2jMqMC>.

- Goldman, Alvin and Bob Beddar (2016). "Reliabilist Epistemology". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Winter 2016. Metaphysics Research Lab, Stanford University.
- Goldman, Alvin I. (1976). "Discrimination and Perceptual Knowledge". In: *Journal of Philosophy* 73.November, pp. 771–791.
- Goldman, Alvin I. (1999). *Knowledge in a Social World*. Oxford University Press.
- Goldstine, Herman H. (1972). *The Computer from Pascal to Von Neumann*. Princeton, NJ, USA: Princeton University Press. ISBN: 0691081042.
- Golik, P., P. Doetsch, and H. Ney (2013). "Cross-Entropy vs. Squared Error Training:a Theoretical and Experimental Comparison". In: *Human Language Technology and Pattern Recognition, Computer Science Department, RWTH Aachen University, 52056 Aachen, Germany*, p. 1.
- Gong, Y. (2013). *Global Operations Strategy: Fundamentals and Practice*. Springer Texts in Business and Economics. Springer Berlin Heidelberg. ISBN: 9783642367083. URL: <https://books.google.de/books?id=GVFBAAAQBAJ>.
- Google-Brain-Team (Nov. 2018). *TensorFlow Webpage*. URL: <https://www.tensorflow.org/> (visited on 11/04/2018).
- Google-Brain-Team (Nov. 2019). *TensorFlowGraph Example Webpage*. URL: <https://www.tensorflow.org/tensorboard/r2/graphs> (visited on 08/27/2019).
- Google-Brain-Team and Community (Nov. 2018a). *Lucid Repository*. URL: <https://github.com/tensorflow/lucid> (visited on 11/10/2018).
- Google-Brain-Team and Community (Nov. 2018b). *TensorFlow Repository*. URL: <https://github.com/tensorflow/tensorflow> (visited on 11/03/2018).
- Goos, G. (1997). *Vorlesungen über Informatik. Bd. I : Grundlagen und funktionales Programmieren*. 2. Berlin: Springer.
- Grau, Alexander (2015). "Prozess". In: *Metzler Lexikon Philosophie: Begriffe und Definitionen*. Ed. by P. Precht and F.P. Burkard. J.B. Metzler, pp. 489–490. ISBN: 9783476054692. DOI: <https://doi.org/10.1007/978-3-476-054692>. URL: <https://books.google.de/books?id=T-5FDwAAQBAJ>.
- Graves, A. (2012). "Supervised Sequence Labelling with Recurrent Neural Networks". In: *Studies in Computational Intelligence* 385, pp. 5–13.
- Graves, A. and J. Schmidhuber (2005). "Framewise phoneme classification with bidirectional LSTM and other neural network architectures". In: *Neural Networks* 18.5–6, pp. 602–610.
- Graves, Alex (2013a). "Generating Sequences With Recurrent Neural Networks". In: *CoRR* abs/1308.0850. arXiv: 1308.0850. URL: <http://arxiv.org/abs/1308.0850>.
- Graves, Alex (2013b). *RNNLIB: A recurrent neural network library for sequence learning problems*. URL: <http://sourceforge.net/projects/rnlib/>.
- Gray, J H and I L Densten (2005). "Towards an integrative model of organizational culture and knowledge management". English. In: *International Journal of Organisational Behaviour* 9.2, pp. 594–603. ISSN: 14405377.
- Greff, Klaus, Rupesh Kumar Srivastava, and Jürgen Schmidhuber (2015). *Brainstorm: Fast, Flexible and Fun Neural Networks, Version 0.5*. URL: <https://github.com/IDSIA/brainstorm>.
- Greff, Klaus et al. (2017). "The Sacred Infrastructure for Computational Research". In: *Proceedings of the 16th Python in Science Conference*. Ed. by Katy Huff et al., pp. 49–56. DOI: <https://doi.org/10.25080/shinma-7f4c6e7008>.

- Grigoryev, Ilya (2018). *AnyLogic 8 in Three Days*. Vol. 5. The AnyLogic Company.
- Grochla, E. (1969). "Modelle als Instrumente der Unternehmensführung". In: *Zeitschrift für betriebswirtschaftliche Forschung* 21, pp. 382–397.
- Gronau, N. (2003). *Wissensmanagement: Potenziale—Konzepte—Werkzeuge*. Reihe Wirtschaftsinformatik. GITO-Verlag. ISBN: 9783936771145. URL: <https://books.google.de/books?id=pUfddYAft1EC>.
- Gronau, N. (2006a). *M-WISE: Modellierung wissensintensiver Prozesse im Software Engineering*. Gito. URL: <https://books.google.de/books?id=-abi4m7aT0C>.
- Gronau, N. (2006b). *Wandlungsfähige Informationssystemarchitekturen—Nachhaltigkeit bei organisatorischem Wandel* (2. Auflage). Reihe Wirtschaftsinformatik. Gito-Verlag. ISBN: 9783936771602. URL: <https://books.google.de/books?id=r7KwU5NZ7wcC>.
- Gronau, N. (2009). *Wissen prozessorientiert managen: Methode und Werkzeuge für die Nutzung des Wettbewerbsfaktors Wissen in Unternehmen*. Oldenbourg Wissenschaftsverlag.
- Gronau, N. (2016). *Geschäftsprozessmanagement in Wirtschaft und Verwaltung*. Gito.
- Gronau, N. (2017). *Geschäftsprozessmanagement in Wirtschaft und Verwaltung*. 2nd. Gito.
- Gronau, N. and M. Grum (2016). *Wissensmanagement im Zeitalter der Digitalisierung*. GITO mbH Verlag. ISBN: 9783955451844. URL: <https://books.google.de/books?id=kbiwAQAAQAAJ>.
- Gronau, N. and M. Grum (2019). *Knowledge Transfer Speed Optimizations in Product Development Contexts*. GITO mbH Verlag Berlin.
- Gronau, N., M. Grum, and B. Bender (2016). "Determining the optimal level of autonomy in cyber-physical production systems". In: *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, pp. 1293–1299.
- Gronau, N., C. Müller, and M. Uslar (2004). "Practical Aspects of Knowledge Management: 5th International Conference, PAKM 2004, Vienna, Austria, December 2–3, 2004. Proceedings". In: ed. by Dimitris Karagiannis and Ulrich Reimer. Berlin, Heidelberg: Springer Berlin Heidelberg. Chap. The KMDL Knowledge Management Approach: Integrating Knowledge Conversions and Business Process Modeling, pp. 1–10. ISBN: 978-3-540-30545-3. DOI: https://doi.org/10.1007/978-3-540-30545-3_1. URL: http://dx.doi.org/10.1007/978-3-540-30545-3_1.
- Gronau, N. et al. (2003). "Modellierung von wissensintensiven Geschäftsprozessen mit der Beschreibungssprache K-Modeler". In: *WM 2003, Professionelles Wissensmanagement—Erfahrungen und Visionen, Luzern*. Ed. by U. Reimer et al., pp. 315–322.
- Gronau, N. et al. (2012). *Modeling and Analyzing knowledge intensive business processes with KMDL. Comprehensive insights into theory and practice*. Ed. by N. Gronau. Gito.
- Gronau, Norbert (2011). "Bessere Geschäftsprozesse mit ERP-Systemen". In: *ERP Management* 2, pp. 26–29.
- Gronau, Norbert and Jane Fröming (2006). "KMDL—Eine semiformale Beschreibungssprache zur Modellierung von Wissenskonversionen". In: *WIRTSCHAFTSINFORMATIK* 48.5, pp. 349–360. ISSN: 18618936. DOI: <https://doi.org/10.1007/s11576-006-0080-4>. URL: <https://doi.org/10.1007/s11576-006-0080-4>.
- Gronau, Norbert and Marcus Grum (2017). "A Visionary Way to Novel Process Optimization Techniques—The Transfer of a Process Modeling Language to the Neuronal Level". In: *Proceedings of the Seventh BMSD*, pp. 11–18.

- Gronau, Norbert and Christiaan Maasdorp (2016). *Modeling of organizational knowledge and information: analyzing knowledge-intensive business processes with KMDL*. GITÖ mbH Verlag Berlin. ISBN: 978-3955451523.
- Gronau, Norbert and Claudia Müller (2005). *Wissensarbeit prozessorientiert modellieren und verbessern*.
- Gronau, Norbert, Claudia Müller, and Roman Korf (Apr. 28, 2005). “KMDL—Capturing, Analysing and Improving Knowledge-Intensive Business Processes”. In: *Journal of Universal Computer Science* 11.4, pp. 452–472.
- Gronau, Norbert and Edzard Weber (2004a). “Management of Knowledge Intensive Business Processes”. In: *Business Process Management*. Ed. by Jörg Desel, Barbara Pernici, and Mathias Weske. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 163–178. ISBN: 978-3-540-25970-1.
- Gronau, Norbert and Edzard Weber (2004b). “Modeling of Knowledge intensive business processes with the declaration language KMDL”. In: *Mehdi Khosrow-Pour (ed.): Innovations Through Information Technology, Proceedings of the 14th Information Resources Management Association International Conference, Idea Group Inc.*
- Gronau, Norbert et al. (2016). “A Proposal to Model Knowledge in Knowledge-Intensive Business Processes”. In: *Proceedings of the Sixth International Symposium on Business Modeling and Software Design Volume 1: BMSD*, INSTICC. SciTePress, pp. 98–103.
- Gruber, Thomas R. (1993). “A translation approach to portable ontology specifications”. In: *Knowledge Acquisition* 5, pp. 199–220.
- Grum, M. (2020). “Managing Human and Artificial Knowledge Bearers—The Creation of a Symbiotic Knowledge Management Approach”. In: *Proceedings of the Tenth BMSD*, pp. 182–201. DOI: https://doi.org/10.1007/978-3-030-24854-3_7.
- Grum, M., B. Bender, and A. Alfa (2017). “The construction of a common objective function for analytical infrastructures”. In: *2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pp. 219–225. DOI: <https://doi.org/10.1109/ICE.2017.8279892>.
- Grum, M. et al. (2018). “A decision maxim for efficient task realization within analytical network infrastructures”. In: *Decision Support Systems*. ISSN: 0167-9236. DOI: <https://doi.org/10.1016/j.dss.2018.06.005>. URL: <http://www.sciencedirect.com/science/article/pii/S0167923618301027>.
- Grum, M. et al. (2019). “Accelerating Knowledge—The Speed Optimization of Knowledge Transfers”. In: *Proceedings of the Ninth BMSD*. DOI: https://doi.org/10.1007/978-3-030-24854-3_7.
- Grum, M. et al. (2020a). “Design of a Neuronal Training Modeling Language Exemplified with the AI-Based Dynamic GUI Adaption”. In: *Wirtschaftsinformatik*.
- Grum, Marcus and Norbert Gronau (2017). “Integration of Augmented Reality Technologies in Process Modeling—The Augmentation of Real World Scenarios With the KMDL”. In: *Proceedings of the Seventh BMSD*, pp. 206–214.
- Grum, Marcus and Norbert Gronau (2018a). “A Visionary Way to Novel Process Optimization Techniques—The Marriage of the Process Domain and Deep Neuronal Networks”. In: *Springer LNBIP*, pp. 1–24.
- Grum, Marcus and Norbert Gronau (2018b). “Process Modeling within the Augmented Reality—The Bidirectional Interplay of Two Worlds”. In: *Proceedings of the Eighth BMSD*, pp. 206–214.

- Grum, Marcus et al. (2020b). "Research Challenges of Knowledge Modeling and the Outline of a Research Agenda". In: *International Forum on Knowledge Asset Dynamics—IFKAD*.
- Gueniche, Ted et al. (2015). "CPT+: Decreasing the Time/Space Complexity of the Compact Prediction Tree". In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Tru Cao et al. Cham: Springer International Publishing, pp. 625–636. ISBN: 978-3-319-18032-8.
- Guha, Subashish, William J. Kettinger, and James T.C. Teng (1993). "BUSINESS PROCESS REENGINEERING". In: *Information Systems Management* 10.3, pp. 13–22. DOI: <https://doi.org/10.1080/10580539308906939>. eprint: <https://doi.org/10.1080/10580539308906939>. URL: <https://doi.org/10.1080/10580539308906939>.
- Häder, M. (2010). *Empirische Sozialforschung: Eine Einführung*. VS Verlag für Sozialwissenschaften. ISBN: 9783531169231. URL: <https://books.google.de/books?id=pAE4j4-8s50C>.
- Halperin, Jeffrey M and Kurt P Schulz (2006). "Revisiting the role of the prefrontal cortex in the pathophysiology of attention-deficit/hyperactivity disorder." In: *Psychological bulletin* 132.4, p. 560.
- Hamacher, K (2006). "Adaptation in stochastic tunneling global optimization of complex potential energy landscapes". In: *Europhysics Letters (EPL)* 74.6, pp. 944–950. DOI: <https://doi.org/10.1209/epl/i2006-10058-0>.
- Hamacher, Kay and W Wenzel (1999). "Scaling behavior of stochastic minimization algorithms in a perfect funnel landscape". In: *Physical Review E* 59.1, p. 938.
- Hamilton, A. G. (1983). *Numbers, Sets and Axioms: The Apparatus of Mathematics*. Cambridge University Press. DOI: <https://doi.org/10.1017/CBO9781139171618>.
- Hammann, P. (1969). "Entscheidungsmodelle in der betriebswirtschaftlichen Theorie". In: *Zeitschrift für betriebswirtschaftliche Forschung* 21, pp. 457–467.
- Hammel, C. (1999). "Generische Spezifikation betrieblicher Anwendungssysteme". PhD thesis. Universität Bamberg. ISBN: 9783826566578. URL: <https://books.google.de/books?id=UAtkAwAACAAJ>.
- Hammer, B. (2000). "On the approximation capability of recurrent neural networks". In: *Neurocomputing* 31.1–4, pp. 107–123.
- Hammer, M. (1996). *Beyond Reengineering: How the Process-Centered Organization Is Changing Our Work and Our Lives*. HarperCollins. ISBN: 9780887307294.
- Hammer, M. and J. Champy (1993). *Reengineering the Corporation: A Manifesto for Business Revolution*. Harper Business. ISBN: 9780887306402. URL: <https://books.google.de/books?id=VpYgWyc16twC>.
- Hammer, Michael (1990). "Reengineering work: don't automate, obliterate". In: *Harvard business review* 68.4, pp. 104–112.
- Han, Shanshan et al. (2018). "Using the tensorflow deep neural network to classify mainland china visitor behaviours in hong kong from check-in data". In: *ISPRS International Journal of Geo-Information* 7.4, p. 158.
- Hanisch, Bastian et al. (2009). "Knowledge management in project environments". In: *Journal of knowledge management*.
- Hare, R. M. (1981). *Moral Thinking: Its Levels, Method, and Point*. Oxford: Oxford University Press.
- Harrington, H.J. (1991). *Business Process Improvement: The Breakthrough Strategy for Total Quality, Productivity, and Competitiveness*. McGraw-Hill Education. ISBN: 9780070267688.

- Harris, R.L. (1999). *Information Graphics: A Comprehensive Illustrated Reference*. BusinessPro collection. Oxford University Press. ISBN: 9780195135329. URL: <https://books.google.de/books?id=LT1RXREvkGIC>.
- Harris, R.L. (2000). *Information Graphics: A Comprehensive Illustrated Reference*. Oxford University Press. URL: <https://books.google.de/books?id=qusmDAAQBAJ>.
- Hart, C. (1998). *Doing a Literature Review: Releasing the Social Science Research Imagination*. Open university set book. SAGE Publications. ISBN: 9780761959755. URL: <https://books.google.de/books?id=FkXvY8oDFdsC>.
- Hartlieb, E. (2002). *Wissenslogistik: Effektives und effizientes Management von Wissensressourcen*. Techno-ökonomische Forschung und Praxis. Deutscher Universitätsverlag. ISBN: 9783824406166. URL: <https://books.google.de/books?id=DTmRAAAACAAJ>.
- Hartree, D.R. (1949). *Calculating instruments and machines*. Univ. of Illinois Press. URL: <https://books.google.de/books?id=VJVRAAAAMAAJ>.
- Hastings, W. K. (1970). "Monte Carlo sampling methods using Markov chains and their applications". In: *Biometrika* 57.1, pp. 97–109. DOI: <https://doi.org/10.1093/biomet/57.1.97>. eprint: <http://biomet.oxfordjournals.org/cgi/reprint/57/1/97.pdf>. URL: <http://biomet.oxfordjournals.org/cgi/content/abstract/57/1/97>.
- Hax, H. (1967). "Bewertungsprobleme bei der Formulierung von Zielfunktionen für Entscheidungsmodelle". In: *Zeitschrift für betriebswirtschaftliche Forschung* 19, pp. 749–761.
- Haykin, Simon (1994). *Neural Networks: A Comprehensive Foundation*. 1st. Upper Saddle River, NJ, USA: Prentice Hall PTR. ISBN: 0023527617.
- Hebb, Donald O. (June 1949). *The organization of behavior: A neuropsychological theory*. New York: Wiley. ISBN: 0-8058-4300-0.
- Hedberg, Bo (1981). "Handbook of organizational design : "how organizations learn and unlearn"". English. In: Bibliography: p. 23–27.
- Heinrich, L.J. and F. Roithmayr (1998). *Wirtschaftsinformatik-Lexikon*. 6. Oldenbourg Verlag München.
- Heinz, K. et al. (1997). *Prozesskostenrechnung für die Logistik kleiner und mittlerer Unternehmen—Methodik und Fallbeispiele*. Praxiswissen, Dortmund.
- Heisig, P. (2002). "Geschäftsprozessorientiertes Wissensmanagement: effektive Wissensnutzung bei der Planung und Umsetzung von Geschäftsprozessen". In: ed. by A. Abdecker et al. Xpert.press. Berlin Heidelberg: Springer Berlin Heidelberg. Chap. GPO-WM: Methode und Werkzeug zum geschäftsprozessorientierten Wissensmanagement, pp. 47–64.
- Heisig, P. (2005). "Integration von Wissensmanagement in Geschäftsprozesse". eureki. Berlin (Dissertation, Technische Universität Berlin)
- Heisig, Peter (1999). "Total Business Knowledge—Spitzenleistung durch Kernkompetenz—Vom Qualitätsmanagement zum Wissensmanagement". In: ed. by J.P. Bläsing, D. Heimann, and E. Högle. TQU Verlag. Chap. Geschäftsprozessorientiertes Wissensmanagement, pp. 27–42.
- Heisig, P. (2000). "Organisation. Schlank—Schnell—Flexibel". In: ed. by R. Bühner. Landsberg, Lech: Verlag mod-erne industrie. Chap. Benchmarking Knowledge Management und wissensorientierte Gestaltung von Geschäftsprozessen, pp. 1–38.
- Heisig, P. (2006). "The GPO-WM® Method for the Integration of Knowledge Management into Business Processes". In: *International Conference on Knowledge Management*. Graz, Austria, pp. 331–337.

- Heisig, P. (2009). "Harmonisation of knowledge management-comparing 160 KM frameworks around the globe". In: *Journal of knowledge management* 13.4, pp. 4–31.
- Hennig, C. et al. (2015). *Handbook of Cluster Analysis*. Chapman & Hall/CRC Handbooks of Modern Statistical Methods. CRC Press. ISBN: 9781466551893. URL: <https://books.google.de/books?id=tDc0CwAAQBAJ>.
- Herbert, M. et al. (2011). *Business Process Management Tools—Ein Marktüberblick*. Fraunhofer Verlag.
- Herrmann, H.-J. (1992). "Modellgestützte Planung in Unternehmen: Entwicklung eines Rahmenkonzepts". PhD thesis. Universität Köln.
- Hertz, John, Richard G. Palmer, and Anders S. Krogh (1991). *Introduction to the Theory of Neural Computation*. 1st. Perseus Publishing. ISBN: 0201515601.
- Herz, Andreas V. M. et al. (2006). "Modeling Single-Neuron Dynamics and Computations: A Balance of Detail and Abstraction". In: *Science* 314.5796, pp. 80–85. ISSN: 0036-8075. DOI: <https://doi.org/10.1126/science.1127240>. eprint: <https://science.scienmag.org/content/314/5796/80.full.pdf>. URL: <https://science.scienmag.org/content/314/5796/80>.
- Hess, T. and L. Brecht (1996). *State of the Art des Business Process Redesign: Darstellung und Vergleich bestehender Methoden*. Gabler Verlag. ISBN: 9783409237864. URL: <https://books.google.de/books?id=iTArPwAACAAJ>.
- Hess, Thomas and Dagmar Schuller (2005). "Business Process Reengineering als nachhaltiger Trend? Eine Analyse der Praxis in deutschen Großunternehmen nach einer Dekade". In: *Schmalenbachs Zeitschrift für betriebswirtschaftliche Forschung* 57.4, pp. 355–373. ISSN: 2366-6153. DOI: <https://doi.org/10.1007/BF03372770>. URL: <https://doi.org/10.1007/BF03372770>.
- Hesse, W. et al. (1994). "Terminologie der Softwaretechnik. Ein Begriffssystem für die Analyse und Modellierung von Anwendungssystemen. Teil 2: Tätigkeits- und ergebnisbezogene Elemente". In: *Informatik Spektrum* 17.2, pp. 96–105.
- Hestenes, M. R. and E. Stiefel (1952). "Methods of conjugate gradients for solving linear systems". In: *Journal of Research of National Bureau of Standards* 49.6, pp. 409–436.
- Hevner, Alan R. et al. (Mar. 2004). "Design Science in Information Systems Research". In: *MIS Q.* 28.1, pp. 75–105. ISSN: 0276-7783. URL: <http://dl.acm.org/citation.cfm?id=2017212.2017217>.
- Heymann, H.W. (1998). "Üben und wiederholen—Neu betrachtet". In: *Pädagogik* 10, pp. 7–11.
- Heyse, V. and J. Erpenbeck (2009). *Kompetenztraining: 64 modulare Informations- und Trainingsprogramme für die betriebliche, pädagogische und psychologische Praxis*. Schäffer-Poeschel. ISBN: 9783791027319. URL: <https://books.google.de/books?id=je0sOwAACAAJ>.
- Hines, Michael L and Nicholas T Carnevale (1997). "The NEURON simulation environment". In: *Neural computation* 9.6, pp. 1179–1209.
- Hinkelmann, Knut, Dimitris Karagiannis, and Rainer Telesko (2002). "PROMOTE—Methodologie und Werkzeug für geschäftsprozessorientiertes Wissensmanagement". In: *Geschäftsprozessorientiertes Wissensmanagement: Effektive Wissensnutzung bei der Planung und Umsetzung von Geschäftsprozessen*. Ed. by Andreas Abecker et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 65–90. ISBN: 978-3-642-55921. DOI: <https://doi.org/10.1007/978-3-642-55921>.

- doi.org/10.1007/978-3-642-55921-1_4. URL: https://doi.org/10.1007/978-3-64255921-1_4.
- Hinkelmann, Knut, Barbara Thönsse, and Fabian Probst (2005). “Referenzmodellierung für E-Government-Services”. In: *Wirtschaftsinformatik* 47.5, pp. 356–366.
- Hintermüller, M., K. Ito, and K. Kunisch (Aug. 2002). “The Primal-Dual Active Set Strategy As a Semismooth Newton Method”. In: *SIAM J. on Optimization* 13.3, pp. 865–888. ISSN: 1052-6234. DOI: <https://doi.org/10.1137/S1052623401383558>. URL: <https://doi.org/10.1137/S1052623401383558>.
- Hinton, Geoffrey E. (2012). “A Practical Guide to Training Restricted Boltzmann Machines.” In: *Neural Networks: Tricks of the Trade* (2nd ed.) Ed. by Grégoire Montavon, Genevieve B. Orr, and Klaus-Robert Müller. Vol. 7700. Lecture Notes in Computer Science. Springer, pp. 599–619. ISBN: 978-3-642-35288-1. URL: <http://dblp.uni-trier.de/db/series/lncs/lncs7700.html#Hinton12>.
- Hinton, Geoffrey E and Ruslan R Salakhutdinov (2012). “A Better Way to Pretrain Deep Boltzmann Machines”. In: *Advances in Neural Information Processing Systems* 25. Ed. by F. Pereira et al. Curran Associates, Inc., pp. 2447–2455. URL: <http://papers.nips.cc/paper/4610-a-better-way-topretrain-deep-boltzmann-machines.pdf>.
- Hochreiter, S. (1991). “Untersuchungen zu dynamischen neuronalen Netzen”. In: *PhD thesis, Institut für Informatik, Technische Universität München*, p. 1. URL: <http://ni.cs.tu-berlin.de/~hochreit/papers/hochreiter.dipl.ps.gz>.
- Hochreiter, S. and J. Schmidhuber (1997a). “Long short-term memory”. In: *Neural computation* 9, pp. 1735–1780. URL: <http://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735?journalCode=neco>.
- Hochreiter, S. et al. (2001). “Gradient flow in recurrent nets: the difficulty of learning long-term dependencies”. In: *IEEE Press, A Field Guide to Dynamical Recurrent Neural Networks, S.C. Kremer and J.F. Kolen, editors*, p. 1.
- Hochreiter, Sepp and Jürgen Schmidhuber (Nov. 1997b). “Long Short-Term Memory”. In: *Neural Comput.* 9.8, pp. 1735–1780. ISSN: 0899-7667. DOI: <https://doi.org/10.1162/neco.1997.9.8.1735>. URL: <http://dx.doi.org/10.1162/neco.1997.9.8.1735>.
- Hodgkin, A. L. and A. F. Huxley (1952). “A quantitative description of membrane current and its application to conduction and excitation in nerve”. In: *The Journal of Physiology* 117.4, pp. 500–544. DOI: <https://doi.org/10.1113/jphysiol.1952.sp004764>. eprint: <https://physoc.onlinelibrary.wiley.com/doi/pdf/10.1113/jphysiol.1952.sp004764>. URL: <https://physoc.onlinelibrary.wiley.com/doi/abs/10.1113/jphysiol.1952.sp004764>.
- Holden, Nigel J. and Harald F. O. Von Kortzfleisch (2004). “Why cross-cultural knowledge transfer is a form of translation in more ways than you think”. In: *Knowledge and Process Management* 11.2, pp. 127–136. DOI: <https://doi.org/10.1002/kpm.198>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/kpm.198>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/kpm.198>.
- Hong, Silien et al. (2012). “Computing a Hierarchical Static Order for Decision Diagram-Based Representation from P/T Nets”. In: *Trans. Petri Nets and Other Models of Concurrency* 5, pp. 121–140.
- Hoover, Stewart V. and Ronald F. Perry (1990). *Simulation: A Problem-Solving Approach*. Addison-Wesley.
- Hopfield, J. J. (1982). “Neural networks and physical systems with emergent collective computational abilities”. In: *PNAS* 79.8, pp. 2554–2558.

- Hornik, K., M. Stinchcombe, and H White (1989). "Multilayer feedforward networks are universal approximators". In: *Neural Network, ISSN 0893-6080* 2.5, pp. 359–366. URL: doi: [https://doi.org/10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8).
- Huber, George P. (Feb. 1991). "Organizational Learning: The Contributing Processes and the Literatures". In: *Organization Science* 2.1, pp. 88–115. ISSN: 1526-5455. DOI: <https://doi.org/10.1287/orsc.2.1.88>. URL: <http://dx.doi.org/10.1287/orsc.2.1.88>.
- Hucka, M. et al. (Mar. 2003). "The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models". In: *Bioinformatics* 19.4, pp. 524–531. ISSN: 1367-4803. DOI: <https://doi.org/10.1093/bioinformatics/btg015>. eprint: <http://oup.prod.sis.lan/bioinformatics/article-pdf/19/4/524/581654/btg015.pdf>. URL: <https://doi.org/10.1093/bioinformatics/btg015>.
- Huettel, S. A., A. W. Song, and G. McCarthy (2009). *Functional Magnetic Resonance Imaging*. (2. ed.), Massachusetts: Sinauer, ISBN 978-0-87893-286-3.
- Huff, A. and D. Chappell (1993). *Party Politics Contribution to Organizational Learning*. Working Paper. Universities of Illinois and Colorado.
- Hughes, Richard I. G. (1997). "Models and Representation". In: *Philosophy of Science* 64.
- Hurwitz, J.S., M. Kaufman, and A. Bowles (2015). *Cognitive Computing and Big Data Analytics*. Wiley. ISBN: 9781118896631. URL: <https://books.google.de/books?id=V41xBgAAQBAJ>.
- IBM-Corporation (1970). *Flowcharting Techniques*. GC20-8152-1. White Plains, N.Y.: IBM Corp.
- IBM-Corporation (1975). *Business systems planning: information systems planning guide*. 1st ed. White Plains GE20-0527-1. NY: International Business Machines Corporation.
- IBM-Corporation (1978). *Business systems planning: information systems planning guide*. 2nd ed. White Plains GE20-0527-2. NY: International Business Machines Corporation.
- IBM-Corporation (1981). *Business systems planning: information systems planning guide*. 3rd ed. White Plains GE20-0527-3. NY: International Business Machines Corporation.
- IBM-Corporation (1984). *Business systems planning: information systems planning guide*. 4th ed. White Plains GE20-0527-4. NY: International Business Machines Corporation.
- Ichikawa, Jonathan Jenkins and Matthias Steup (2018a). "The Analysis of Knowledge". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Summer 2018. Metaphysics Research Lab, Stanford University.
- Ichikawa, Jonathan Jenkins and Matthias Steup (2018b). "The Analysis of Knowledge". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Summer 2018. Metaphysics Research Lab, Stanford University.
- IDSIA, CogBotLab (Nov. 2009). *PyBrain v0.3 Documentation*. URL: <http://pybrain.org/docs/tutorial/intro.html> (visited on 11/12/2009).
- IDSIA, CogBotLab (Nov. 2018). *PyBrain Repository*. URL: <https://github.com/pybrain> (visited on 11/03/2018).
- Iivari, Juhani, Rudy Hirschheim, and Heinz K. Klein (2004). "Towards a distinctive body of knowledge for Information Systems experts: coding ISD process knowledge in two IS journals". In: *Information Systems Journal* 14.4, pp. 313–342. DOI: <https://doi.org/10.1111/j.1365-2575.2004.00177.x>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1365-2575.2004.00177.x>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2575.2004.00177.x>.

- Imboden Dieter, M. Imboden and Sabine Koch (2003). *Systemanalyse: Einführung in die mathematische Modellierung natürlicher Systeme*. 1st ed. Springer-Lehrbuch. Springer-Verlag Berlin Heidelberg. ISBN: 9783-642-62878-8, 978-3-642-55667-8. URL: <http://gen.lib.rus.ec/book/index.php?md5=94fad44100bc9595aecf071cb8ca15b2>.
- Inkpen, Andrew C. (2008). "Knowledge transfer and international joint ventures: the case of NUMMI and General Motors". In: *Strategic Management Journal* 29.4, pp. 447–453. DOI: <https://doi.org/10.1002/smj.663>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/smj.663>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/smj.663>.
- Irvine, Andrew David (2015). "Alfred North Whitehead". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Winter 2015. Metaphysics Research Lab, Stanford University.
- Ishikawa, Kaoru (1985). *What is total quality control? The Japanese way*. Prentice Hall.
- ISO Central Secretary (1985). *Information processing—Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts*. en. Standard ISO/IEC 5807:1985. Geneva, CH: International Organization for Standardization. URL: <https://www.iso.org/standard/11955.html>.
- ISO Central Secretary (2003). *Information technology—SGML applications—Topic maps*. en. Standard ISO/IEC 13250:2003. Geneva, CH: International Organization for Standardization.
- ISO Central Secretary (2005). *Information technology—Open Distributed Processing—Unified Modeling Language (UML) Versionen 1.4.2*. en. Standard ISO/IEC 19501:2005. Geneva, CH: International Organization for Standardization. URL: <https://www.iso.org/standard/32620.html>.
- ISO Central Secretary (2016). *Information technology—Mathematical Markup Language (MathML) Version 3.0 2nd Edition*. en. Standard ISO/IEC 40314:216. Geneva, CH: International Organization for Standardization.
- ISO/IEC-International-Standard (1990). *Information Resource Dictionary System (IRDS)—Framework ISO/EIC 10027*. Geneva: International Organization for Standardization.
- IUPHAR and BPS (Aug. 2019). A Collection of Neuropeptides. URL: <https://www.guidetopharmacology.org/GRAC/FamilyDisplayForward?familyId=965>.
- Jackson, Cecile (1983). *System Development*.
- Jackson, M. A. (1975). *Principles of Program Design*. Orlando, FL, USA: Academic Press, Inc. ISBN: 0123790506.
- Jacobson, I., M. Ericsson, and A. Jacobson (1995). *The Object Advantage: Business Process Reengineering with Object Technology*. ACM Press books. Addison-Wesley. ISBN: 9780201422894.
- Jacobson, Ivar (1992). *Object-oriented Software Engineering*. New York, NY, USA: ACM. ISBN: 0-201-54435-0.
- Jain, A. K., Jianchang Mao, and K. M. Mohiuddin (1996). "Artificial neural networks: a tutorial". In: *Computer* 29.3, pp. 31–44. ISSN: 0018-9162. DOI: <https://doi.org/10.1109/2.485891>.
- James, JQ, Albert YS Lam, and Victor OK Li (2011). "Evolutionary artificial neural network based on chemical reaction optimization". In: *2011 IEEE Congress of Evolutionary Computation (CEC)*. IEEE, pp. 2083–2090.
- Jammer, M. (1965). "Die Entwicklung des Modellbegriffes in den physikalischen Wissenschaften". In: *Studium Generale* 18.3, p. 166.173.

- Jarke, Matthias and Stefanie Kethers (1999). "Regionale Kooperationskompetenz: Probleme und Modellierungstechniken". In: *Wirtschaftsinformatik* 41.4, pp. 316–325. URL: <http://dblp.uni-trier.de/db/journals/wi/wi41.html#JarkeK99>.
- Jeffcoat, David E and Robert L Bulfin (1993). "Simulated annealing for resource-constrained scheduling". In: *European Journal of Operational Research* 70.1, pp. 43–51.
- Jelinek, M. (1979). *Institutionalizing Innovation: A Study of Organizational Learning Systems*. Praeger special studies. Praeger. URL: https://books.google.de/books?id=_WxAAAAIAAJ.
- Jenni, Urs and Andreas Ziltener (2007). "Conceptual Framework for an R&D Cooperation Model in SMEs". In: *Indian Journal of Economics & Business* Special Issue (2008), pp. 53–72.
- Jensen, Kurt (1996). *Coloured Petri Nets—Basic Concepts, Analysis Methods and Practical Use—Volume 1, Second Edition*. Monographs in Theoretical Computer Science. An EATCS Series. Springer. ISBN: 978-3-64208243-6. DOI: <https://doi.org/10.1007/978-3-662-03241-1>. URL: <https://doi.org/10.1007/978-3-662-03241-1>.
- Jessen K.R, Mirsky R. (1980). "Glial cells in the enteric nervous system contain glial fibrillary acidic protein". In: *Nature*.
- Jeusfeld, Manfred A. and Uwe A. Johnen (1995). "An executable meta model for re-engineering of database schemas". In: *International journal of cooperative information systems: IJCIS* 4, pp. 237–258. ISSN: 0218-2157. URL: <http://publications.rwth-aachen.de/record/136096>.
- Jia, Yangqing et al. (2014). "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *CoRR* abs/1408.5093. arXiv: 1408.5093. URL: <http://arxiv.org/abs/1408.5093>.
- Jiroveanu, G., B. De Schutter, and R.K. Boel (Jan. 2007). "The on-line diagnosis of time Petri nets based on partial orders". In: *Taming Heterogeneity and Complexity of Embedded Control*. Ed. by F. Lamnabhi-Lagarrigue et al. London, UK: ISTE. Chap. 21, pp. 363–391.
- Jones, Neil D., Lawrence H. Landweber, and Y. Edmund Lien (1977). "Complexity of Some Problems in Petri Nets." In: *Theoretical Computer Science* 4.3, pp. 277–299. DOI: [https://doi.org/10.1016/0304-3975\(77\)900147](https://doi.org/10.1016/0304-3975(77)900147). URL: <http://dblp.uni-trier.de/db/journals/tcs4.html#JonesLL77>.
- Jones, Nicola (Jan. 2014). "The Learning Machines". In: *Nature* 505, pp. 146–148. URL: <http://www.nature.com/news/computer-science-the-learning-machines-1.14481>.
- Jones, Teresa, W. Roy Schulte, and Michele Cantara (2014). *Magic Quadrant for Intelligent Business Process Management Suite*. Tech. rep. G00255421. Gartner Inc.
- Joost, M. and W. Schiffmann (1998). "Speeding up backpropagation algorithms by using crossentropy combined with pattern normalization". In: *International Journal of Uncertainty, Fuzziness and Knowledge-BasedSystems* 6.2, pp. 117–126.
- Juengst, K.L. and P. Strittmatter (1995). "Wissensstrukturdarstellung. Theoretische Ansätze und praktische Relevanz". In: *Unterrichtswissenschaft* 23, pp. 194–207.
- Jung, Martin and Linder Hoehe (2015). "Creating and validating a microscopic pedestrian simulation to analyze an airport security checkpoint". In: *Proceedings of the 2015 Winter Simulation Conference*. L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti.

- Junginger, Stefan et al. (2000). "Ein geschäftsprozessmanagement-werkzeug der nächsten generation—ADONIS: konzeption und anwendungen". In: *Wirtschaftsinformatik* 42.5, pp. 392–401.
- Justus, A. (1999). *Wissenstransfer in strategischen Allianzen: eine verhaltenstheoretische Analyse*. Institut für Unternehmensführung Göttingen: Schriften des Instituts für Unternehmensführung der Georg-August-Universität Göttingen. Lang. ISBN: 9783631342831. URL: <https://books.google.de/books?id=JPGIAAAACAAJ>.
- Kaas, Jon H et al. (1979). "Multiple representations of the body within the primary somatosensory cortex of primates". In: *Science* 204.4392, pp. 521–523.
- Kaastra, Ibeling and Milton Boyd (1996). "Designing a neural network for forecasting financial". In: *Neurocomputing* 10, pp. 215–236.
- Kain, Alexander (2010). "Wikinomics (Don Tapscott, Anthony D. Williams)". In: *Social Media Handbuch*. Nomos Verlagsgesellschaft mbH & Co. KG, pp. 149–161.
- Kalisch, Dominik P.H. (2003). *SIMS—über die Möglichkeiten der Simulation gruppendynamischer Prozesse*. Tectum.
- Känel, W. (1966). *Operations Research und betriebswirtschaftliche Entscheidungen*. Hamburg: von Decker.
- Kanter, I. and H. Sompolinsky (1987). "Associative recall of memory without errors". In: *Phys. Rev. A* 35 (1), pp. 380–392. DOI: <https://doi.org/10.1103/PhysRevA.35.380>. URL: <https://link.aps.org/doi/10.1103/PhysRevA.35.380>.
- Kaplan, R. and D. Norton (1992). "The Balanced Scorecard—Measures That Drive Performance". In: *Harvard Business Review* 79.
- Kaplan, R and David Norton (1993). "9/1/1993 Putting the Balanced Scorecard to Work". In: *Harvard Business Review*, pp. 134–47.
- Karagiannis, D. and R. Telesko (2000). "The EU-Project PROMOTE: A Process-oriented Approach for Knowledge Management". In: *Proc. of the Third Int. Conf. of Practical Aspects of Knowledge Management*, pp. 9–18.
- Karagiannis, D. and R. Woitsch (2002). "Wissensmanagement: Strategien—Prozesse—Communities". In: ed. by N. Gronau. Aachen: Shaker Verlag Aachen. Chap. The Promote approach: Modelling Knowledge Management Processes to describe Knowledge Management Systems, pp. 35–52.
- Karagiannis, Dimitris (Aug. 1995). "BPMS: Business Process Management Systems". In: *SIGOIS Bull.* 16.1, pp. 10–13. ISSN: 0894-0819. DOI: <https://doi.org/10.1145/209891.209894>. URL: <http://doi.acm.org/10.1145/209891.209894>.
- Karagiannis, Dimitris, Stefan Junginger, and Robert Strobl (1996). "Introduction to Business Process Management Systems Concepts". In: *Business Process Modelling*. Ed. by Bernd Scholz-Reiter and Eberhard Stickel. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 81–106. ISBN: 978-3-642-80317-8.
- Kasabov, N.K. (1996). *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering*. Bradford book. MIT Press. ISBN: 9780262112123. URL: <https://books.google.de/books?id=9bdwtUQLchIC>.
- Keller, G., M. Nüttgens, and A. W. Scheer (1992). *Semantische Prozessmodellierung auf der Grundlage „Ereignisgesteuerter Prozessketten (EPK)“*. 89. Institut für Wirtschaftsinformatik, Universität Saarbrücken.

- Keller, G. and T. Teufel (1997). *SAP R 3 prozessorientiert anwenden: iteratives Prozess-Prototyping zur Bildung von Wertschöpfungsketten*. Edition SAP. Addison-Wesley. ISBN: 9783827310736. URL: <https://books.google.de/books?id=VSPnPQAAQAAJ>.
- Kellner, Marc and Gregory Hansen (1988). *Software Process Modeling*. Tech. rep. CMU/SEI-88-TR-009. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. URL: <http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=10619>.
- Kennedy, M. P. and L. O. Chua (1988). “Neural networks for nonlinear programming”. In: *IEEE Transactions on Circuits and Systems* 35.5, pp. 554–562. ISSN: 0098-4094. DOI: <https://doi.org/10.1109/31.1783>.
- Kern, W. (1964). *Operations Research: Eine Einführung in die Optimierungskunde*. Stuttgart: Poeschel.
- Kernighan, Brian W. and Rob Pike (1984). *Unix programming environment*. Prentice Hall.
- Kethers, S. (2000). “Multi-Perspective Modeling and Analysis of Cooperation Processes”. PhD thesis. Von Der Fakultät Für Mathematik et al.
- Kim, D. (1992). *Systems Archetypes*. Toolbox Reprint Series. Waltham, MA: Pegasus Communications.
- Kim, D.H. and V. Anderson (1998). *Systems Archetype Basics: From Story to Structure*. Systems thinking tools—the Pegasus workbook series. Pegasus Communications. ISBN: 9781883823184. URL: <https://books.google.de/books?id=xF96PQAAQAAJ>.
- Kim, Phil (2017). “Deep Learning”. In: *MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence*. Berkeley, CA: Apress, pp. 103–120. ISBN: 978-1-4842-2845-6. DOI: https://doi.org/10.1007/9781-4842-2845-6_5. URL: https://doi.org/10.1007/9781-4842-2845-6_5.
- Kindermans, Pieter-Jan et al. (2017a). “Learning how to explain neural networks: Patternnet and patternattribution”. In: *arXiv preprint arXiv:1705.05598*.
- Kindermans, Pieter-Jan et al. (2017b). “The (Un)reliability of saliency methods.” In: *CoRR* abs/1711.00867. URL: <http://dblp.uni-trier.de/db/journals/corr/corr1711.html#abs-1711-00867>.
- Kindermans, Pieter-Jan et al. (2018). “Learning how to explain neural networks: PatternNet and PatternAttribution”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=Hkn7CBaTW>.
- Kiran, D.R. (2016). *Total Quality Management: Key Concepts and Case Studies*. Elsevier Science. URL: <https://books.google.de/books?id=PIIkDAAAQBAJ>.
- Kirk, G.S., J.E. Raven, and M. Schofield (1983). *The Presocratic Philosophers: A Critical History with a Selection of Texts*. Cambridge University Press. ISBN: 9780521274555. URL: <https://books.google.de/books?id=kFpd86J8PLsC>.
- Kirkham, Richard L. (1984). “Does the Gettier Problem Rest on a Mistake?” In: *Mind* 93.372, pp. 501–513.
- Kirkpatrick, Scott, C Daniel Gelatt, and Mario P Vecchi (1983). “Optimization by simulated annealing”. In: *science* 220.4598, pp. 671–680.
- Kitchenham, Barbara et al. (Jan. 2009). “Systematic Literature Reviews in Software Engineering—A Systematic Literature Review”. In: *Inf. Softw. Technol.* 51.1, pp. 7–15. ISSN: 0950-5849. DOI: <https://doi.org/10.1016/j.infsof.2008.09.009>. URL: <http://dx.doi.org/10.1016/j.infsof.2008.09.009>.

- Klaus, G. et al. (1968). *Wörterbuch der Kybernetik*. Hrsg. von G. Klaus. [Autoren und Mitarbeiter: O. Bär, J. Behr, M. Bierwisch u.a.]. Dietz. URL: <https://books.google.de/books?id=LrO4jwEACAAJ>.
- Klein, J. (1991). "Darstellung der Problematik heterogener betrieblicher Informationssysteme am Informationsmodell der Unternehmung". In: *IM Information Management* 4.
- Klein, Jonathan I. (1989). "Parenthetic Learning in Organizations: Toward the Unlearning of the Unlearning Model". In: *Journal of Management Studies* 26.3, pp. 291–308. DOI: <https://doi.org/10.1111/j.1467-6486.1989.tb00729.x>.
- Klimecki, R. G. and M. Thomae (1997). *Organisationales Lernen*. Tech. rep. 18. University of Konstanz. URL: <https://d-nb.info/981158110/34>.
- Kline, D. M. and V. L. Berardi (2005). "Revisiting squared-error and cross-entropy functions for training neural network classifiers". In: *Neural Computing And Applications*, Springer 14, Issue 4, pp. 310–318.
- KMDL Blog (June 2019). *Modelangelo*. URL: <http://www.modelangelo.com> (visited on 2016).
- Knapp, H.-G. (1978). "Zur Semantik quantitativer Modelle". In: *Quantitative Ansätze in der Betriebswirtschaftslehre: Bericht von der wissenschaftlichen Tagung des Verbandes der Hochschullehrer für Betriebswirtschaft e.V. vom 1.–3. Juni 1977 in Darmstadt*. Ed. by H. Müller-Merbach. München: Vahlen, pp. 199–213.
- Knapper, Rico et al. (2010). "PROSA—Process-Oriented Service Level Agreements for Providing a Single Point of Contract". In: *MKWI*. Universitätsverlag Göttingen, pp. 219–231.
- Knolmayer, Gerhard, Rainer Endl, and Marcel Pfahrer (2000). "Modeling Processes and Workflows by Business Rules". In: *Business Process Management: Models, Techniques, and Empirical Studies*. Ed. by Wil van der Aalst, Jörg Desel, and Andreas Oberweis. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 16–29. ISBN: 978-3-540-45594-3. DOI: https://doi.org/10.1007/3-540-45594-9_2. URL: https://doi.org/10.1007/3540-45594-9_2.
- Knothe, K. and H. Wessels (1999). *Finite Elemente: eine Einführung für Ingenieure*. Springer. URL: https://books.google.de/books?id=2k6SvcO_3ogC.
- Knudsen, E I, S Lac, and S D Esterly (1987). "Computational Maps in the Brain". In: *Annual Review of Neuroscience* 10.1. PMID: 3551761, pp. 41–65. DOI: <https://doi.org/10.1146/annurev.ne.10.030187.000353>. eprint: <https://doi.org/10.1146/annurev.ne.10.030187.000353>. URL: <https://doi.org/10.1146/annurev.ne.10.030187.000353>.
- Kofman, Fred and Peter M. Senge (1993). "Communities of commitment: The heart of learning organizations". In: *Organizational Dynamics* 22.2, pp. 5–23. ISSN: 0090-2616. DOI: [https://doi.org/10.1016/0090-2616\(93\)90050-B](https://doi.org/10.1016/0090-2616(93)90050-B). URL: <http://www.sciencedirect.com/science/article/pii/009026169390050B>.
- Kohonen, T. (1989). "Self-organization and associative memory". In: *Springer-Verlag New York, Inc., New York, NY, USA, ISBN 0-387-51387-6* 3rd edition, p. 1.
- Koornneef, F. (2000). "Organised Learning from Small-scale Incidents". PhD thesis. Delft University.
- Koornneef, F. and A.R. Hale (2004). "How to Manage Experience Sharing—from Organisational Surprises to Organisational Knowledge". In: ed. by J.H. Andriessen and B Fahlbruch. Amsterdam: Elsevier Science. Chap. Organisational Learning and Theories of Action.

- Kosiol, E. (1961). *Modellanalyse als Grundlage unternehmerischer Entscheidungen*. Westdeutscher. URL: <https://books.google.de/books?id=urh2oAEACAAJ>.
- Kosko, Bart (1992). *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc. ISBN: 0-13-612334-1.
- Kottemann, Jeffrey E. and Benn R. Konsynski (1984). "Dynamic Metasystems for Information Systems Development". In: *ICIS*.
- Kotusev, Svyatoslav (2016). "The History of Enterprise Architecture: An Evidence-Based Review". In: *Journal of Enterprise Architecture* 12.1, pp. 29–37.
- Krallmann, H., A Bobrik, and O. Levina (2013). *Systemanalyse im Unternehmen—Prozessorientierte Methoden der Wirtschaftsinformatik*. Oldenbourg Verlag München.
- Krallmann, H., H. Frank, and N. Gronau (2002). *Systemanalyse im Unternehmen*. Vol. 4. Oldenbourg Verlag München Wien.
- Kramer, Ulrich and Mihaela Neculau (1998). *Simulationstechnik*. Carl-Hanser Verlag.
- Krcmar, H. (2015). *Informationsmanagement*. Springer Berlin Heidelberg. ISBN: 9783662458631. URL: <https://books.google.de/books?id=tEv0BwAAQBAJ>.
- Krcmar, H. and P. Elgass (1993). "Team und Informationsmanagement". In: *Handbuch Informationsmanagement: Aufgaben—Konzepte—Praxislösungen*. Ed. by A.W. Scheer. Gabler Verlag. ISBN: 9783409199384. URL: <https://books.google.de/books?id=j6UdAQAAQAAJ>.
- Kreikebaum, H. (1993). *Strategische Unternehmensplanung*. Vol. 5. Stuttgart.
- Kriwet, C.K. (1997). *Inter-and Intraorganizational Knowledge Transfer*. Dissertationen / Universität St. Gallen. na. URL: <https://books.google.de/books?id=zhNYNAEACAAJ>.
- Krogh, G., K. Ichijo, and I. Nonaka (2000). *Enabling Knowledge Creation: How to Unlock the Mystery of Tacit Knowledge and Release the Power of Innovation*. Oxford University Press.
- Kruse, R. et al. (2015). *Computational Intelligence: Eine methodische Einführung in Künstliche Neuronale Netze, Evolutionäre Algorithmen, Fuzzy-Systeme und Bayes-Netze*. 2nd ed. Computational Intelligence. Springer Fachmedien Wiesbaden. ISBN: 9783658109035.
- Kubat, Miroslav (Dec. 1998). "Decision trees can initialize radial-basis function networks". English. In: *IEEE Transactions on Neural Networks* 9.5, pp. 813–821. ISSN: 1045-9227. DOI: <https://doi.org/10.1109/72.712154>.
- Kubicek, Herbert (2002). "Wie kann eDemocracy Bürgerbeteiligung erweitern?" In: *Fachzeitschrift des Kompetenzzentrums eGovernment* 2.
- Kubicek, Herbert, Hilmar Westholm, and Martin Wind (2002). "Wahlen und Bürgerbeteiligung via Internet". In: *HMD—Praxis Wirtschaftsinform.* 226.
- Kuckartz, U. (2014). *Mixed Methods: Methodologie, Forschungsdesigns und Analyseverfahren*. Springer Fachmedien Wiesbaden. ISBN: 9783531932675. URL: <https://books.google.de/books?id=iQJYBAAQBAJ>.
- Kueng, P. et al. (1996). "How to compose an Object-Oriented Business Process Model?" In: *Method Engineering: Principles of method construction and tool support*. Ed. by Sjaak Brinkkemper, Kalle Lyytinen, and Richard J. Welke. Boston, MA: Springer US, pp. 94–110. ISBN: 978-0-387-35080-6. DOI: https://doi.org/10.1007/9780-387-35080-6_7. URL: https://doi.org/10.1007/978-0-387-35080-6_7.
- Kueng, Peter and Peter Kawalek (1997). "Goalbased business process models: creation and evaluation". In: *Business Process Management Journal* 3.1, pp. 17–38. DOI: <https://doi.org/10.1007/BF02595102>.

- [org/10.1108/14637159710161567](https://doi.org/10.1108/14637159710161567). eprint: <https://doi.org/10.1108/14637159710161567>. URL: <https://doi.org/10.1108/14637159710161567>.
- Kuhlang, P., K.W. Wagner, and A. Moser (2010). *Geschäftsprozessmanagement-Tools: Marktstudie und Auswahlunterstützung*. NWV—Neuer Wiss. Verlag. ISBN: 9783708307268. URL: <https://books.google.de/books?id=FaKzXwAACAAJ>.
- Kuhlen, R., T. Seeger, and D. Strauch (2004). *Grundlagen der praktischen Information und Dokumentation*. Grundlagen der praktischen Information und Dokumentation Bd. 1. K.G. Saur. ISBN: 9783598116742. URL: <https://books.google.de/books?id=A18VAQAAIAAJ>.
- Kulhavy, E. (1963). *Operations Research: Die Stellung der Operationsforschung in der Betriebswirtschaftslehre*. Wiesbaden: Gabler.
- Küll, R. and P. Stähly (2013). *Simulation als betriebliche Entscheidungshilfe: State of the Art und neuere Entwicklungen*. Berlin Heidelberg: Springer.
- Kunze, Matthias and Mathias Weske (2010). “Signavio-Oryx Academic Initiative”. In: *Proceedings of the Business Process Management 2010 Demonstration Track*, Hoboken, NJ, USA, September 14–16, 2010. URL: <http://ceur-ws.org/Vol-615/paper6.pdf>.
- Lachmann, Rolf-Jürgen (2015). “Prozessphilosophie”. In: *Metzler Lexikon Philosophie: Begriffe und Definitionen*. Ed. by P. Precht and F.P. Burkard. J.B. Metzler, pp. 490–491. ISBN: 9783476054692. DOI: <https://doi.org/10.1007/978-3-476-05469-2>. URL: <https://books.google.de/books?id=T-5FDwAAQBAJ>.
- Lachs, C.P.P.J., J. Lachs, and R.B. Talisse (2008). *American Philosophy: An Encyclopedia*. Taylor & Francis. ISBN: 9781135948870. URL: <https://books.google.de/books?id=oP6TAgAAQBAJ>.
- Lam, Alice (1997). “Embedded Firms, Embedded Knowledge: Problems of Collaboration and Knowledge Transfer in Global Cooperative Ventures”. In: *Organization Studies* 18.6, pp. 973–996. DOI: <https://doi.org/10.1177/017084069701800604>. eprint: <https://doi.org/10.1177/017084069701800604>. URL: <https://doi.org/10.1177/017084069701800604>.
- Lämmel, U. and J. Cleve (2012). *Künstliche Intelligenz*. München: Carl-Hanser Verlag.
- Lang, Michael and Jim Duggan (2001). “A Tool to Support Collaborative Software Requirements Management”. In: *Requirements Engineering* 6.3, pp. 161–172. ISSN: 1432-010X. DOI: <https://doi.org/10.1007/s007660170002>. URL: <https://doi.org/10.1007/s007660170002>.
- Lass, Sander (2017). “Simulationskonzept zur Nutzenvalidierung cyber-physischer Systeme in komplexen Fabrikumgebungen”. PhD thesis. University of Potsdam.
- Lau, Adela and Eric Tsui (2009). “Knowledge management perspective on e-learning effectiveness”. In: *Knowledge-based systems* 22.4, pp. 324–325.
- Laukemann, Alexander, Hansgeorg Binz, and Daniel Roth (2017). “Concept for a simulation model to analyze knowledge conversions within the product development process”. In: *DS 87-6 Proceedings of the 21st International Conference on Engineering Design (ICED17), Vol. 6: Design Information and Knowledge*. Ed. by Anja Maier et al. Vol. 6. ICED Series. Design Society, pp. 21–30.
- Law, A.M. and W.D. Kelton (1991). *Simulation Modeling and Analysis*. McGraw-Hill series in industrial engineering and management science. McGraw-Hill. ISBN: 9780070366985. URL: <https://books.google.de/books?id=jefQAAAAMAAJ>.
- Law, A.M. and W.D. Kelton (2000). *Simulation modeling and analysis*. McGraw-Hill series in industrial engineering and management science. McGraw-Hill. ISBN: 9780070592926. URL: <https://books.google.de/books?id=QqkZAQAAIAAJ>.

- Le, Quoc V. et al. (2011). "Building high-level features using large scale unsupervised learning". In: *CoRR* abs/1112.6209. URL: <http://arxiv.org/abs/1112.6209>.
- LeCun, Y. et al. (1998). "Neural Networks: Tricks of the trade". In: *Efficient Backprop*. In G.Orr and M.K., editors, Springer, p. 1.
- Lederer, Albert L. and Veronica Gardiner (1992). "STRATEGIC INFORMATION SYSTEMS PLANNING". In: *Information Systems Management* 9.3, pp. 13–20. DOI: <https://doi.org/10.1080/10580539208906877>. eprint: <https://doi.org/10.1080/10580539208906877>. URL: <https://doi.org/10.1080/10580539208906877>.
- Lederer, Albert L. and Andrew G. Putnam (1986). "Connecting Systems Objectives to Business Strategy with BSP". In: *Information Strategy: The Executives' Journal* 2.2, pp. 12–18.
- Lederer, Albert L. and Andrew G. Putnam (1987). "Bridging the Gap: Connecting Systems Objectives to Business Strategy with BSP". In: *Journal of Information Systems Management* 4.3, pp. 40–46. DOI: <https://doi.org/10.1080/07399018708962858>. eprint: <https://doi.org/10.1080/07399018708962858>. URL: <https://doi.org/10.1080/07399018708962858>.
- Lee, Maria R. and Yi-Chen Lan (2011). "Toward a unified knowledge management model for SMEs". In: *Expert Systems with Applications* 38.1, pp. 729–735. ISSN: 0957-4174. DOI: <https://doi.org/10.1016/j.eswa.2010.07.025>. URL: <http://www.sciencedirect.com/science/article/pii/S0957417410006421>.
- Leedy, P.D. and J.E. Ormrod (2005). *Practical Research: Planning and Design*. Prentice Hall. ISBN: 9780131108950. URL: <https://books.google.de/books?id=MipiQgAACAAJ>.
- LEGO Group (Sept. 2013a). *Lego Color Sorter Instruction*. URL: <https://le-www-live-s.legocdn.com/sc/media/lessons/mindstorms-ev3/building-instructions/ev3-model-coreset-color-sorter-c778563f88c986841453574495cb5ff1.pdf> (visited on 2013).
- LEGO Group (Sept. 2013b). *LEGO MINDSTORMS Education EV3 Set*. URL: <https://education.lego.com/dede/products/lego-mindstorms-education-ev3-set/5003400#product>.
- LEGO Group (Sept. 2013c). *Lego Robot Arm Instruction*. URL: <https://le-www-live-s.legocdn.com/sc/media/lessons/mindstorms-ev3/building-instructions/ev3-model-core-setrobot-arm-h25-56cdb22c1e3a02f1770bda72862ce2bd.pdf> (visited on 2013).
- Lehner, F. (1995). "Modelle und Modellierung". In: *Wirtschaftsinformatik: theoretische Grundlagen*. Ed. by F. Lehner, K. Hildebrand, and R. Maier. München: Hanser, pp. 73–164. ISBN: 9783446180024. URL: <https://books.google.de/books?id=PWV3AAAACAAJ>.
- Lehner, F. (2013). "Modelle und Modellierung in der Wirtschaftsinformatik". In: *Selbstverständnis betriebswirtschaftlicher Forschung und Lehre: Tagung der Kommission Wissenschaftstheorie*. Ed. by H. Wächter. Gabler Verlag, pp. 55–86. ISBN: 9783663130574. URL: <https://books.google.de/books?id=7RfNBgAAQBAJ>.
- Lehner, F. (2014). *Wissensmanagement: Grundlagen, Methoden und technische Unterstützung*. Carl Hanser Verlag GmbH & Company KG. ISBN: 9783446441521.
- Lehner, F., K. Hildebrand, and R. Maier (1995). *Wirtschaftsinformatik: theoretische Grundlagen*. Hanser. ISBN: 9783446180024. URL: <https://books.google.de/books?id=PWV3AAAACAAJ>.
- Lehrer, Keith and Thomas Paxson Jr (1969). "Knowledge: Undefeated Justified True Belief". In: *Journal of Philosophy* 66.8, pp. 225–237.
- Lengerich, Benjamin J et al. (2017). "Towards visual explanations for convolutional neural networks via input resampling". In: *arXiv preprint arXiv: 1707.09641*.

- Levinthal, Daniel A. (Feb. 1991). "Organizational Adaptation and Environmental Selection-Interrelated Processes of Change". In: *Organization Science* 2.1, pp. 140–145. ISSN: 1526-5455. DOI: <https://doi.org/10.1287/orsc.2.1.140>. URL: <http://dx.doi.org/10.1287/orsc.2.1.140>.
- Levitt, Barbara and James G. March (1988). "Organizational Learning". In: *Annual Review of Sociology* 14.1, pp. 319–338. DOI: <https://doi.org/10.1146/annurev.so.14.080188.001535>. eprint: <https://doi.org/10.1146/annurev.so.14.080188.001535>. URL: <https://doi.org/10.1146/annurev.so.14.080188.001535>.
- Levy, Y. and T. J. Ellis (2006). "A Systems Approach to Conduct an Effective Literature Review in Support of Information Systems Research". In: *Informing Science Journal* 9. Ed. by E. Cohen, pp. 181–212. URL: <http://inform.nu/Articles/Vol9/V9p181-212Levy99.pdf>.
- Liberati, Alessandro et al. (July 2009). "The PRISMA Statement for Reporting Systematic Reviews and Meta-Analyses of Studies That Evaluate Health Care Interventions: Explanation and Elaboration". In: *PLOS Medicine* 6.7, pp. 1–28. DOI: <https://doi.org/10.1371/journal.pmed.1000100>. URL: <https://doi.org/10.1371/journal.pmed.1000100>.
- Liebelt, W. and M. Sulzberger (1992). *Grundlagen der Ablauforganisation*. Vol. 2. Gießen.
- Liebl, F. (1992a). *Simulation: problemorientierte Einführung*. Vol. 1. Oldenbourg. ISBN: 9783486233735. URL: <https://books.google.de/books?id=zfd4AAAACAAJ>.
- Liebl, F. (1992b). *Simulation: problemorientierte Einführung*. Vol. 2. Oldenbourg. ISBN: 9783486233735. URL: <https://books.google.de/books?id=zfd4AAAACAAJ>.
- Liebowitz, Jay (1999). "Key ingredients to the success of an organization's knowledge management strategy". In: *Knowledge and Process Management* 6.1, pp. 37–40. DOI: [https://doi.org/10.1002/\(SICI\)1099-1441\(199903\)6:1<37::AID-KPM40>3.0.CO;2-M](https://doi.org/10.1002/(SICI)1099-1441(199903)6:1<37::AID-KPM40>3.0.CO;2-M). eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/%28SICI%291099-1441%28199903%29%3A1%3C37%3A%3AAID-KPM40%3E3.0.CO%3B2-M>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291099-1441%28199903%29%3A1%3C37%3A%3AAID-KPM40%3E3.0.CO%3B2-M>.
- Light, R.A. (Sept. 2020). *Mosquitto Message Broker*. URL: <https://mosquitto.org/>.
- Light, Roger A. (2017). "Mosquitto: server and client implementation of the MQTT protocol". In: *Journal of Open Source Software* 2.13, p. 265. DOI: <https://doi.org/10.21105/joss.00265>. URL: <https://doi.org/10.21105/joss.00265>.
- Lin, Hsiu? Fen (2007). "Knowledge sharing and firm innovation capability: an empirical study". In: *International Journal of Manpower* 28.3/4, pp. 315–332. DOI: <https://doi.org/10.1108/01437720710755272>. eprint: <https://doi.org/10.1108/01437720710755272>. URL: <https://doi.org/10.1108/01437720710755272>.
- Lin, Min, Qiang Chen, and Shuicheng Yan (2013). "Network in network". In: *arXiv preprint arXiv*: 1312.4400.
- LISA lab, University of Montreal and Theano Development Team (Nov. 2018a). *Theano Repository*. URL: <https://github.com/Theano/Theano> (visited on 11/03/2018).
- LISA lab, University of Montreal and Theano Development Team (Nov. 2018b). *Theano Webpage*. URL: <http://deeplearning.net/software/theano/#> (visited on 11/04/2018).
- List, B. and B. Korherr (2006). "An Evaluation of Conceptual Business Process Modelling Languages". In: *Proceedings of the 2006 ACM Symposium on Applied Computing*. SAC '06. Dijon, France: ACM, pp. 1532–1539. ISBN: 1-59593-108-2. DOI: <https://doi.org/10.1145/1141277.1141633>. URL: <http://doi.acm.org/10.1145/1141277.1141633>.

- Lloyd, Catherine M., Matt D.B. Halstead, and Poul F. Nielsen (2004a). “CellML: its future, present and past”. In: *Progress in Biophysics and Molecular Biology* 85.2. Modelling Cellular and Tissue Function, pp. 433–450. ISSN: 0079-6107. DOI: <https://doi.org/10.1016/j.biophys.2004.01.004>. URL: <http://www.sciencedirect.com/science/article/pii/S007961070400015X>.
- Lloyd, Catherine M., Matt D.B. Halstead, and Poul F. Nielsen (2004b). “CellML: its future, present and past”. In: *Progress in Biophysics and Molecular Biology* 85.2. Modelling Cellular and Tissue Function, pp. 433–450. ISSN: 0079-6107. DOI: <https://doi.org/10.1016/j.biophys.2004.01.004>. URL: <http://www.sciencedirect.com/science/article/pii/S007961070400015X>.
- Loo, Sandra K. et al. (2009). “Cortical activity patterns in ADHD during arousal, activation and sustained attention”. In: *Neuropsychologia* 47.10, pp. 2114–2119. ISSN: 0028-3932. DOI: <https://doi.org/10.1016/j.neuropsychologia.2009.04.013>. URL: <http://www.sciencedirect.com/science/article/pii/S0028393209001638>.
- Lübbe, Alexander, Thomas Allweyer, and Sven Schnägelberger (2015). *BPM Toolmarktmonitor 2015*. Tech. rep. BPM&O.
- Lübbe, Alexander and Sven Schnägelberger (2014). *BPM Toolmarktmonitor 2014*. Tech. rep. BPM&O.
- Lübbe, Alexander and Sven Schnägelberger (2016). *BPM Toolmarktmonitor 2016*. Tech. rep. BPM&O.
- Lübbe, Alexander and Sven Schnägelberger (2018). *BPM Toolmarktmonitor: BPM in the Cloud*. Tech. rep. BPM&O.
- Lübbe, Alexander, Sven Schnägelberger, and Mike Pelz (2016). *BPM Toolmarktmonitor 2016—Kostenlose Werkzeuge für die Prozessmodellierung*. Tech. rep. BPM&O.
- Lübke, J. et al. (2003). “Morphometric analysis of the columnar innervation domain of neurons connecting layer 4 and layer 2/3 of juvenile rat barrel cortex.” In: *Cereb Cortex* 10.
- Lufi, Dubi, Susan Okasha, and Arie Cohen (2004). “Test Anxiety and its Effect on the Personality of Students with Learning Disabilities”. In: *Learning Disability Quarterly* 27.3, pp. 176–184. DOI: <https://doi.org/10.2307/1593667>. eprint: <https://doi.org/10.2307/1593667>. URL: <https://doi.org/10.2307/1593667>.
- Luhmann, N. (1984). *Soziale Systeme: Grundriss einer allgemeinen Theorie*. Suhrkamp. ISBN: 9783518577004. URL: <https://books.google.de/books?id=13cSAQAAQAAJ>.
- Luong, Minh-Thang et al. (2015). “Multi-task sequence to sequence learning”. In: *arXiv preprint arXiv: 1511.06114*. URL: <http://dblp.uni-trier.de/db/journals/corr/corr1511.html#LuongLSVK15>.
- Lutterer, Wolfram (2012). “Deutero-learning”. In: *Encyclopedia of the Sciences of Learning*. Ed. by Norbert M. Seel. Boston, MA: Springer US, pp. 939–941. ISBN: 978-1-4419-1428-6. DOI: https://doi.org/10.1007/978-1-44191428-6_260. URL: https://doi.org/10.1007/978-1-4419-1428-6_260.
- Lutz, M. (2008). *Steuerung internationaler Forschungs- und Entwicklungsnetzwerke*. Logos Verlag Berlin. ISBN: 9783832521172. URL: <https://books.google.de/books?id=gYgwDI7yeW8C>.
- Lyles, Marjorie A. and Charles R. Schwenk (1992). “TOP MANAGEMENT, STRATEGY AND ORGANIZATIONAL KNOWLEDGE STRUCTURES”. In: *Journal of Management Studies* 29.2, pp. 155–174. DOI: <https://doi.org/10.1111/j.1467-6486.1992.tb00658.x>.

- MacGill, M.E. and J.W. Slocum (1996). *Das intelligente Unternehmen: Wettbewerbsvorteile durch schnelle Anpassung an Marktbedürfnisse*. Schäffer-Poeschel. ISBN: 9783791009834. URL: <https://books.google.de/books?id=y4mSAAACAAJ>.
- MacIntosh, R. and A. Francis (1997). *The Market, Technological and Industry Contexts of Business Process Re-Engineering in UK Business*. Research Report. Department of Management Studies, Glasgow Business School.
- MacMenamin, S.M. and J.F. Palmer (1988). *Strukturierte Systemanalyse*. Hanser. ISBN: 9783446151666. URL: <https://books.google.de/books?id=ACoSKwAACAAJ>.
- Mahalanobis, Prasanta Chandra (1936). "On the generalized distance in statistics". In: *Proceedings of the National Institute of Sciences (Calcutta)* 2, pp. 49–55.
- Mahendran, Aravindh and Andrea Vedaldi (2015). "Understanding deep image representations by inverting them." In: *CVPR*. IEEE Computer Society, pp. 5188–5196. ISBN: 978-1-4673-6964-0. URL: <http://dblp.uni-trier.de/db/conf/cvpr/cvpr2015.html#MahendranV15>.
- Mahmoud, Mohammed et al. (2009). "A formal framework for scenario development in support of environmental decision-making". In: *Environmental Modelling & Software* 24.7, pp. 798–808. ISSN: 1364-8152. DOI: <https://doi.org/10.1016/j.envsoft.2008.11.010>. URL: <http://www.sciencedirect.com/science/article/pii/S1364815208002211>.
- Maier, R. (2007). *Knowledge Management Systems: Information and Communication Technologies for Knowledge Management*. Springer Berlin Heidelberg. ISBN: 9783540714088. URL: <https://books.google.de/books?id=r5pAHIN1ChwC>.
- Maier, R., T. Hädrich, and R. Peinl (2005). *Enterprise Knowledge Infrastructures*. Springer Berlin Heidelberg. ISBN: 9783540239154. URL: <https://books.google.de/books?id=eo5h7DulR04C>.
- Malek, Alaeddin and Maryam Yashtini (2009). "A Neural Network Model for Solving Non-linear Optimization Problems with Real-Time Applications". In: *Advances in Neural Networks—ISNN 2009*. Ed. by Wen Yu, Haibo He, and Nian Zhang. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 98–108. ISBN: 978-3-642-01513-7.
- Mandel, H and F Fischer (2000). *Wissen sichtbar machen. Wissensmanagement mit Mappingtechniken*.
- Mandl, Heinz and Gabi Reinmann-Rothmeier (2000). "Individuelles Wissensmanagement: Strategien für den persönlichen Umgang mit Information und Wissen am Arbeitsplatz". In: *Bern, Göttingen, Toronto, Seattle: Huber*.
- March, J. G. and J. P. Olsen (1975). "The Uncertainty of the Past: Organizational Learning Under Ambiguity". In: *European Journal of Political Research* 3.2, pp. 147–171. DOI: <https://doi.org/10.1111/j.1475-6765.1975.tb00521.x>.
- March, J.G. and J.P. Olsen (1976). *Ambiguity and choice in organizations*. Universitetsforlaget. URL: https://books.google.de/books?id=_mVHAAAAMAAJ.
- Marsan, Marco Ajmone et al. (1994). *Modelling with Generalized Stochastic Petri Nets*. 1st. New York, NY, USA: John Wiley & Sons, Inc. ISBN: 0471930598.
- Marsland, Stephen, Jonathan Shapiro, and Ulrich Nehmzow (2002). "A self-organising network that grows when required". In: *Neural Networks* 15.8, pp. 1041–1058. ISSN: 0893-6080. DOI: [https://doi.org/10.1016/S0893-6080\(02\)00078-3](https://doi.org/10.1016/S0893-6080(02)00078-3). URL: <http://www.sciencedirect.com/science/article/pii/S0893608002000783>.
- Martin, J. (1982). *Strategic Data-Planning Methodologies*. Englewood Cliffs, N.J.: Yourdon Press.

- Martinetz, T. M., S. G. Berkovich, and K. J. Schulten (July 1993). “‘Neural-gas’ Network for Vector Quantization and Its Application to Time-series Prediction”. In: *Trans. Neur. Netw.* 4.4, pp. 558–569. ISSN: 1045-9227. DOI: <https://doi.org/10.1109/72.238311>. URL: <https://doi.org/10.1109/72.238311>.
- Martinetz, Thomas and Klaus Schulten (Mar. 1994). “Topology Representing Networks”. In: *Neural Netw.* 7.3, pp. 507–522. ISSN: 0893-6080. DOI: [https://doi.org/10.1016/0893-6080\(94\)90109-0](https://doi.org/10.1016/0893-6080(94)90109-0). URL: [http://dx.doi.org/10.1016/0893-6080\(94\)90109-0](http://dx.doi.org/10.1016/0893-6080(94)90109-0).
- Martinetz, Thomas M. and Klaus J. Schulten (1991). “A ‘Neural Gas’ Network Learns Topologies”. In: *Proceedings of the International Conference on Artificial Neural Networks 1991* (Espoo, Finland). Ed. by Teuvo Kohonen et al. Amsterdam; New York: North-Holland, pp. 397–402.
- Masaaki, Imai (1986). “Kaizen: The key to Japan’s competitive success”. In: *New York, Ltd: McGraw-Hill*.
- Mass, N. (1980). “Elements of the System Dynamics Method”. In: ed. by J. Renders. Waltham, MA: Pegasus Communications. Chap. Stock and flow variables and the dynamics of supply and demand.
- MathWorks, The (2019). *Matlab Webpage*. URL: <https://de.mathworks.com/products/matlab.html> (visited on 01/04/2019).
- Matilal, Bimal Krishna (1986). *Perception: An Essay on Classical Indian Theories of Knowledge*. Oxford University Press.
- Matloff, Norm (2008). “Introduction to discrete-event simulation and the simpy language”. In: *Davis, CA. Dept of Computer Science. University of California at Davis. Retrieved on August 2.2009*, pp. 1–33.
- Matthes, D. (2011). *Enterprise Architecture Frameworks Kompendium: Über 50 Rahmenwerke für das IT-Management*. Xpert.press. Springer Berlin Heidelberg. ISBN: 9783642129551. URL: <https://books.google.de/books?id=SpQm6oDrnN0C>.
- Mayer, Richard J. and Paula S. DeWitte (1999). “Delivering Results: Evolving BPR from Art to Engineering”. In: *Business Process Engineering: Advancing the State of the Art*. Ed. by D. Jack Elzinga, Thomas R. Gullledge, and Chung-Yee Lee. Boston, MA: Springer US, pp. 83–129. ISBN: 978-1-4615-5091-4. DOI: https://doi.org/10.1007/978-1-4615-5091-4_5. URL: https://doi.org/10.1007/978-1-4615-5091-4_5.
- Mayntz, R. (1967). “Modellkonstruktion—Ansatz, Typen und Zweck”. In: *Formalisierte Modelle in der Soziologie*. Soziologische Texte. Luchterhand. URL: <https://books.google.de/books?id=DdUsAQAAIAAJ>.
- McCulloch, W. S. and W. Pitts (1988). “A logical calculus of the ideas immanent in nervous activity”. In: *MIT Press, Cambridge, MA, USA, ISBN 0-262-01097-6*, pp. 15–27.
- McCulloch, Warren S. and Walter Pitts (1943). “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133. ISSN: 1522-9602. DOI: <https://doi.org/10.1007/BF02478259>. URL: <https://doi.org/10.1007/BF02478259>.
- McEliece, Robert J. et al. (1987). “The capacity of the Hopfield associative memory”. In: *IEEE Trans. Information Theory* 33.4, pp. 461–482. DOI: <https://doi.org/10.1109/TIT.1987.1057328>. URL: <https://doi.org/10.1109/TIT.1987.1057328>.
- McFarlane, DA (2011). “Personal knowledge management (PKM): are we really ready?” In: *Journal of Knowledge Management Practice* 12.3, pp. 108–114.

- McGill, Michael E. and John W. Slocum (1993). "Unlearning the organization". In: *Organizational Dynamics* 22.2, pp. 67–79. ISSN: 0090-2616. DOI: [https://doi.org/10.1016/0090-2616\(93\)90054-5](https://doi.org/10.1016/0090-2616(93)90054-5). URL: <http://www.sciencedirect.com/science/article/pii/0090261693900545>.
- Meel, Jeroen W van, Pieter WG Bots, and Henk G Sol (1994). "Towards a research framework for business engineering". In: *Proceedings of the IFIP TC8 Open Conference on Business Process Re-engineering: Information Systems Opportunities and Challenges*. Elsevier Science Inc., pp. 581–592.
- Meinsen, S. (2003). *Konstruktivistisches Wissensmanagement: wie Wissensarbeiter ihre Arbeit organisieren*. System und Organisation. Beltz. ISBN: 9783407320445. URL: https://books.google.tn/books?id=_eP1wAEACAAJ.
- Mendling, Jan, Gustaf Neumann, and Markus Nüttgens (2005). "Yet another event-driven process chain". In: *International Conference on Business Process Management*. Springer, pp. 428–433.
- Mentzas, Gregoris et al. (2001). "Knowledge networking: a holistic solution for leveraging corporate knowledge". In: *Journal of knowledge management*.
- Mertens, Kai (2001). *Knowledge Management: Best Practices in Europe*. Ed. by Peter Heisig and Jens Vorbeck. <http://publica.fraunhofer.de/documents/N-4420.html>. Berlin, Heidelberg: Springer-Verlag.
- Mertens, Peter (1982). *Simulation*. Vol. 2. Poeschel.
- Mertens, Peter (2010). "Gestaltungsorientierte Wirtschaftsinformatik: Ein Plädoyer für Rigor und Relevanz". In: ed. by Hubert Österle, Robert Winter, and Walter Brenner. Eigenverlag. Chap. Anspruchsguppen der gestaltungsorientierten Wirtschaftsinformatik, pp. 19–25.
- Mertins, Kai, Ina Kohl, and Ronald Orth (2016). "Ein Referenzmodell für Wissensmanagement". In: *Wissensmanagement im Mittelstand: Grundlagen—Lösungen—Praxisbeispiele*. Ed. by Holger Kohl, Kai Mertins, and Holger Seidel. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 31–40. ISBN: 978-3-662-49220-8. DOI: https://doi.org/10.1007/978-3-662-49220-8_4. URL: https://doi.org/10.1007/978-3-66249220-8_4.
- Metropolis, Nicholas et al. (1953). "Equation of State Calculations by Fast Computing Machines". In: *The Journal of Chemical Physics* 21.6, pp. 1087–1092. DOI: <https://doi.org/10.1063/1.1699114>. eprint: <https://doi.org/10.1063/1.1699114>. URL: <https://doi.org/10.1063/1.1699114>.
- Michalewicz, Zbigniew and Marc Schoenauer (Mar. 1996). "Evolutionary Algorithms for Constrained Parameter Optimization Problems". In: *Evol. Comput.* 4.1, pp. 1–32. ISSN: 1063-6560. DOI: <https://doi.org/10.1162/evco.1996.4.1.1>. URL: <http://dx.doi.org/10.1162/evco.1996.4.1.1>.
- Michie, D., D. J. Spiegelhalter, and C. C. Taylor, eds. (1994). *Machine Learning, Neural and Statistical Classification*. Series in Artificial Intelligence. Hemel Hempstead, Hertfordshire, England: Ellis Horwood.
- Mietzner, Dana and Martin Kamprath (2013). "A Competence Portfolio for Professionals in the Creative Industries". In: *Creativity and Innovation Management* 22.3, pp. 280–294. DOI: <https://doi.org/10.1111/caim.12026>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/caim.12026>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/caim.12026>.

- Mildenberger, O. and T. Sauerbier (1999). *Theorie und Praxis von Simulationssystemen: Eine Einführung für Ingenieure und Informatiker*. Studium Technik. Vieweg+Teubner Verlag. ISBN: 9783528038663. URL: <https://books.google.de/books?id=4jpIAAAACAAJ>.
- Mildenberger, O. and T. Sauerbier (2013). *Theorie und Praxis von Simulationssystemen: Eine Einführung für Ingenieure und Informatiker*. Studium Technik. Vieweg+Teubner Verlag. ISBN: 9783322907738. URL: <https://books.google.de/books?id=63PLBgAAQBAJ>.
- Mills, Harlan D., Richard C. Linger, and Alan R. Hevner (1986). *Principles of Information Systems Analysis and Design*. San Diego, CA, USA: Academic Press Professional, Inc. ISBN: 0-12-497545-3.
- Minbaeva, D et al. (2003). "MNC knowledge transfer, subsidiary absorptive capacity, and HRM". In: *Journal of International Business Studies* 34.6, pp. 586–599. URL: <https://EconPapers.repec.org/RePEc:pal:jintbs:v:34:y:2003:i:6:p:586-599>.
- Minh, David D. L. and Do Le (Paul) Minh (2015). "Understanding the Hastings Algorithm". In: *Communications in Statistics—Simulation and Computation* 44.2, pp. 332–349. DOI: <https://doi.org/10.1080/03610918.2013.777455>. eprint: <https://doi.org/10.1080/03610918.2013.777455>. URL: <https://doi.org/10.1080/03610918.2013.777455>.
- Mitchell, Tom M (1997). "Artificial neural networks". In: *Machine learning* 45, pp. 81–127.
- Moen, Ronald and Clifford Norman (2006). *Evolution of the PDCA cycle*.
- Moher, David et al. (July 2009). "Preferred Reporting Items for Systematic Reviews and Meta-Analyses: The PRISMA Statement". In: *PLOS Medicine* 6.7, pp. 1–6. DOI: <https://doi.org/10.1371/journal.pmed.1000097>. URL: <https://doi.org/10.1371/journal.pmed.1000097>.
- Molière, F. de (1984). "Prinzipien des Modellentwurfs: Eine modelltheoretische und gestaltungsoorientierte Betrachtung". PhD thesis. Darmstadt: Technische Hochschule Darmstadt, Fachbereich I, Rechts-und Wirtschaftswissenschaften.
- Montavon, Grégoire et al. (2017). "Explaining nonlinear classification decisions with deep taylor decomposition". In: *Pattern Recognition* 65, pp. 211–222.
- Moody, John E. and Christian J. Darken (1989). "Fast Learning in Networks of Locally-Tuned Processing Units". In: *Neural Computation* 1.2, pp. 281–294. DOI: <https://doi.org/10.1162/neco.1989.1.2.281>. URL: <https://doi.org/10.1162/neco.1989.1.2.281>.
- Moore's laboratory, Duke University and NEURON Development Team (Nov. 2018). *NEURON Webpage*. URL: <https://neuron.yale.edu/neuron/> (visited on 11/18/2018).
- Mordvintsev, A. (Jan. 2015). *DeepDreaming with TensorFlow*. URL: <https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/tutorials/deepdream/deepdream.ipynb> (visited on 08/14/2019).
- Mordvintsev, A., C. Olah, and M. Tyka (June 2015). *Inceptionism: Going Deeper into Neural Networks*. URL: <https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html> (visited on 08/14/2019).
- Moritz, P. et al. (Nov. 2015). "SparkNet: Training Deep Networks in Spark". In: *ArXiv e-prints*. arXiv: 1511.06051 [stat.ML].
- Mountcastle, Vernon B. (1978). "An Organizing Principle for Cerebral Function: The Unit Model and the Distributed System". In: *The Mindful Brain*. Ed. by Gerald M. Edelman and Vernon V. Mountcastle. Cambridge, MA: MIT Press, pp. 7–50.
- Müller, G.H. (1965). "Der Modellbegriff in der Mathematik". In: *Studium Generale* 18.3, pp. 154–165.
- Muller, Klaus and Tony Vignaux (2003). "SimpPy: Simulating systems in python". In: *ONLamp. com Python Devcenter* 650.

- Müller, R. (1983). "Zur Geschichte des Modelldenkens und des Modellbegriffs". In: *Modelle—Konstruktion der Wirklichkeit*. Ed. by H. Stachowiak. Kritische Information. Fink, pp. 17–86. URL: <https://books.google.de/books?id=qfc3AQAAIAAJ>.
- Myers, Glenford J. (1978). *Composite Structure Design*. New York, NY, USA: John Wiley & Sons, Inc. ISBN: 0442805845.
- Myler, H.R. (1998). *Fundamentals of Engineering Programming with C and Fortran*. Cambridge University Press. ISBN: 9780521629508. URL: <https://books.google.de/books?id=lsfMsdBe2IC>.
- Nance, Richard E. (1994). "The Conical Methodology and the evolution of simulation model development". In: *Annals of Operations Research* 53.1, pp. 1–45. ISSN: 1572-9338. DOI: <https://doi.org/10.1007/BF02136825>. URL: <https://doi.org/10.1007/BF02136825>.
- Nauck, D., F. Klawonn, and R. Kruse (1997). *Foundations of Neuro-Fuzzy Systems*. Wiley. ISBN: 97804717971511. URL: <https://books.google.de/books?id=lcvQAAAAMAAJ>.
- Nauck, D. et al. (2013). *Neuro-Fuzzy-Systeme: Von den Grundlagen künstlicher Neuronaler Netze zur Kopplung mit Fuzzy-Systemen*. Computational Intelligence. Vieweg+Teubner Verlag. ISBN: 9783322803368. URL: <https://books.google.de/books?id=WtIIBgAAQBAJ>.
- Nauck, Detlef and Rudolf Kruse (1992). "A Neural Fuzzy Controller Learning by Fuzzy Error Propagation". In: *In Proc. NAFIPS'92*, pp. 388–397.
- Nauck, Detlef and Rudolf Kruse (1993). "A Fuzzy Neural Network Learning Fuzzy Control Rules and Membership Functions by Fuzzy Error Backpropagation". In: *In Proc. IEEE Int. Conf. on Neural Networks*, pp. 1022–1027.
- Neumann, Frank (2015). "Process Elements of Mechatronic Product Development". In: *Analyzing and Modeling Interdisciplinary Product Development: A Framework for the Analysis of Knowledge Characteristics and Design Support*. Wiesbaden: Springer Fachmedien Wiesbaden, pp. 141–155. ISBN: 978-3-658-11092-5. DOI: https://doi.org/10.1007/978-3-658-11092-5_7. URL: https://doi.org/10.1007/978-3-658-11092-5_7.
- Neumann, John Von, A. W. Taub, and A. H. Taub (1963). *The Collected Works of John Von Neumann: 6-Volume Set*. Reader's Digest Young Families. ISBN: 0080095666.
- neuroConstruct-Team et al. (Nov. 2018a). *NeuroConstruct Repository*. URL: <https://github.com/NeuralEnsemble/neuroConstruct> (visited on 11/11/2018).
- neuroConstruct-Team et al. (Nov. 2018b). *NeuroConstruct Webpage*. URL: <http://www.neuroconstruct.org/> (visited on 11/11/2018).
- NEURON-Community (Nov. 2018). *NEURON Repository*. URL: <https://github.com/neuronsimulator/nrn> (visited on 11/18/2018).
- Neuweg, G.H. (1999). *Könnerschaft und implizites Wissen*. Waxmann Verlag. ISBN: 9783830957539. URL: <https://books.google.de/books?id=uuyorC5LSOnoC>.
- Nevis, Edwin C., Anthony J. DiBella, and Janet M. Gould (1995). "Understanding Organizations as Learning Systems". In: *Sloan Management Review* 36.2, pp. 73–85.
- Nguyen, A., J. Yosinski, and J. Clune (2015). "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 427–436. DOI: <https://doi.org/10.1109/CVPR.2015.7298640>.
- Nguyen, Anh et al. (2016a). "Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space". In: *CoRR* abs/1612.00005. arXiv: 1612.00005. URL: <http://arxiv.org/abs/1612.00005>.

- Nguyen, Anh et al. (2016b). "Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space". In: *CoRR* abs/1612.00005. [arXiv: 1612.00005](https://arxiv.org/abs/1612.00005). URL: <http://arxiv.org/abs/1612.00005>.
- Nguyen, Anh Mai et al. (2016c). "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks". In: *CoRR* abs/1605.09304. [arXiv: 1605.09304](https://arxiv.org/abs/1605.09304). URL: <http://arxiv.org/abs/1605.09304>.
- Niedermann, Florian, Sylvia Radeschütz, and Bernhard Mitschang (2011). "Business Process Optimization Using Formalized Optimization Patterns". In: *Business Information Systems*. Ed. by Witold Abramowicz. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 123–135.
- Niedermeyer, Ernst and Fernando L. da Silva (Nov. 2004). *Electroencephalography: Basic Principles, Clinical Applications, and Related Fields*. 5th. Lippincott Williams & Wilkins. ISBN: 0781751268. URL: <http://www.worldcat.org/isbn/0781751268>.
- Nikolai, Cynthia and Gregory Madey (2009). "Tools of the Trade: A Survey of Various Agent Based Modeling Platforms". In: *Journal of Artificial Societies and Social Simulation* 12.2, p. 2. ISSN: 1460-7425. URL: <http://jasss.soc.surrey.ac.uk/12/2/2.html>.
- Nissen, Hans W. (1997). "Separierung und Resolution multipler Perspektiven in der konzeptuellen Modellierung". Zugl.: Aachen, Techn. Hochsch., Diss., 1997. PhD thesis. Sankt Augustin, XII, 273 S. ISBN: 3-89601-438-2. URL: <http://publications.rwth-aachen.de/record/57862>.
- Nissen, Hans W. and Matthias Jarke (1999). "Repository support for multi-perspective requirements engineering". In: *Information Systems* 24.2. Meta-Modelling and Methodology Engineering, pp. 131–158. ISSN: 0306-4379. DOI: [https://doi.org/10.1016/S0306-4379\(99\)00009-5](https://doi.org/10.1016/S0306-4379(99)00009-5). URL: <http://www.sciencedirect.com/science/article/pii/S0306437999000095>.
- Nissen, M., M. Kamel, and K. Sengupta (2000). "Integrated Analysis and Design of Knowledge Systems and Processes". In: *Information Resources Management Journal* 13.1, pp. 24–43.
- Nolan, R.L. and D.W. Mulryan (1987). "Undertaking an Architecture Program". In: *Stage by Stage* 7.2, pp. 1–10.
- Nonaka, Ikujiro and Hirotaka Takeuchi (1995). *The knowledge-creating company: How Japanese companies create the dynamics of innovation*. Oxford university press.
- North, K. (1998). *Wissensorientierte Unternehmensführung: Wertschöpfung durch Wissen*. Gabler. URL: <https://books.google.de/books?id=sFEAAgAACAAJ>.
- North, K., A. Brandner, and M.S. Thomas Steininger (2016). *Wissensmanagement für Qualitätsmanager: Erfüllung der Anforderungen nach ISO 9001:2015*. essentials. Gabler Verlag, Springer Fachmedien Wiesbaden. ISBN: 9783658112493. URL: <http://www.springer.com/de/book/9783658112493>.
- Norton, David (2011). *Magic Quadrant for Intelligent Business Process Management Suite*. Tech. rep. G00219247. Gartner Inc.
- Novák, V., I. Perfilieva, and J. Mockor (1999). *Mathematical Principles of Fuzzy Logic*. The Springer International Series in Engineering and Computer Science. Springer US. ISBN: 9780792385950. URL: <https://books.google.de/books?id=pJeu6Ue65S4C>.
- Nüttgens, Markus and Frank J. Rump (2002). "Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK)". In: *Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen—Promise 2002, 9.–11. Oktober 2002, Potsdam*, pp. 64–77. URL: <http://subs.emis.de/LNI/Proceedings/Proceedings21/article557.html>.

- Nyhuis, P., F. Wriggers, and T. Busse (2008). "Identifikation von Potenzialen in der unternehmensinternen Lieferkette". In: *Industrie Management 2*.
- Nystrom, Paul C. and William H. Starbuck (1984). "To avoid organizational crises, unlearn". In: *Organizational Dynamics* 12.4, pp. 53–65. ISSN: 0090-2616. DOI: [https://doi.org/10.1016/0090-2616\(84\)90011-1](https://doi.org/10.1016/0090-2616(84)90011-1). URL: <http://www.sciencedirect.com/science/article/pii/0090261684900111>.
- Object Management Group (Dec. 2013). *Business Process Model and Notation (BPMN)*—Version 2.0.2. Tech. rep. formal-2013-12-09. Object Management Group. URL: <http://www.omg.org/spec/BPMN>.
- Object Management Group (June 2015). *XML Metadata Interchange (XMI)*—Version 2.5.1. Tech. rep. formal-2015-06-07. Object Management Group. URL: <https://www.omg.org/spec/UML>.
- Object Management Group (May 2017). *Unified Modeling Language (UML)*—Version 2.5.1. Tech. rep. formal-2017-12-05. Object Management Group. URL: <https://www.omg.org/spec/UML>.
- Object Management Group (Jan. 2019). *BPMN Webpage*. URL: <https://www.omg.org/spec/BPMN/2.0> (visited on 01/27/2019).
- O'Brien, E. (2010). *Knowledge Management for Process, Organizational and Marketing Innovation: Tools and Methods: Tools and Methods*. Advances in Knowledge Acquisition, Transfer, and Management: Information Science Reference. ISBN: 9781615208302. URL: <https://books.google.de/books?id=v2tMmRxROS8C>.
- Oja, Erkki (1982). "Simplified neuron model as a principal component analyzer". In: *Journal of Mathematical Biology* 15.3, pp. 267–273. ISSN: 1432-1416. DOI: <https://doi.org/10.1007/BF00275687>. URL: <https://doi.org/10.1007/BF00275687>.
- Olah, Chris, Alexander Mordvintsev, and Ludwig Schubert (2017). "Feature Visualization". In: *Distill*. <https://distill.pub/2017/feature-visualization>. DOI: <https://doi.org/10.23915/distill.00007>.
- Olah, Chris et al. (2018). "The Building Blocks of Interpretability". In: *Distill*. <https://distill.pub/2018/buildingblocks>. DOI: <https://doi.org/10.23915/distill.00010>.
- Olesnitz, Dietrich von der and Michael W Busch (2008). "Die Bedeutung transaktiver Gedächtnissysteme für die Informationsproduktion in Teams". In: *Journal of Business Economics* 78.4, pp. 367–396.
- Olle, T. William, Henk G. Sol, and Ian G. MacDonald (1991). *Information Systems Methodologies; A Framework for Understanding*, 2Nd Ed. 2nd. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0201544431.
- ONNX-Community. *ONNX Webpage*. URL: <https://onnx.ai/> (visited on 12/30/2018).
- O'Reilly, Randall C. and Michael J. Frank (Feb. 2006). "Making Working Memory Work: A Computational Model of Learning in the Prefrontal Cortex and Basal Ganglia". In: *Neural Comput.* 18.2, pp. 283–328. ISSN: 0899-7660. DOI: <https://doi.org/10.1162/089976606775093909>. URL: <http://dx.doi.org/10.1162/089976606775093909>.
- Orr, K. (1971). "Structured Systems Development". New York, NY: *Yourdon Press*.
- Ortner, Erich (1997). *Methodenneutraler Fachentwurf: Zu den Grundlagen einer anwendungsorientierten Informatik*. Teubner-Reihe Wirtschaftsinformatik. Stuttgart, Leipzig: Teubner. URL: <http://tubiblio.ulb.tu-darmstadt.de/48080/>.

- Ossimitz, G. (2000). *Entwicklung systemischen Denkens*. Klagenfurter Beiträge zur Didaktik der Mathematik. Profil. ISBN: 9783890194943. URL: <https://books.google.de/books?id=MLyOgAACAAJ>.
- Österle, Hubert et al. (2010). "Gestaltungsorientierte Wirtschaftsinformatik: Ein Plädoyer für Rigor und Relevanz". In: ed. by Hubert Österle, Robert Winter, and Walter Brenner. Eigenverlag. Chap. Memorandum zur gestaltungsorientierten Wirtschaftsinformatik, pp. 1–6.
- Osterweil, L. (1987). "Software Processes Are Software Too". In: *Proceedings of the 9th International Conference on Software Engineering*. ICSE '87. Monterey, California, USA: IEEE Computer Society Press, pp. 2–13. ISBN: 0-89791-216-0. URL: <http://dl.acm.org/citation.cfm?id=41765.41766>.
- Otte, Sebastian, Dirk Krechel, and Marcus Liwicki (2013). "JANNLab Neural Network Framework for Java". In: *Poster Proceedings Conference MLDM 2013*. New York, USA: ibai-publishing, pp. 39–46.
- Ould, M.A. (1995). *Business Processes: Modelling and Analysis for Re-Engineering and Improvement*. Wiley. ISBN: 9780471953524.
- Oygard, A.M. (July 2015). *Visualizing GoogLeNet Classes*. URL: <https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/tutorials/deepdream/deepdream.ipynb> (visited on 08/14/2019).
- Padberg, Julia (2018). "Subtyping for Hierarchical, Reconfigurable Petri Nets". In: *CoRR abs/1802.04698*.
- Page-Jones, Meilir (1988). *Practical guide to structured systems design*. English. 2nd ed. Includes index. Englewood Cliffs, N.J.: Yourdon Press. ISBN: 0136907695.
- Park, G. et al. (2017). "A modeling framework for business process reengineering using big data analytics and a goal-orientation". In: *2017 11th International Conference on Research Challenges in Information Science (RCIS)*, pp. 21–32.
- Parsons, Talcott (1937). *The structure of social action: a study in social theory with special reference to a group of recent European writers*/by Talcott Parsons. English. McGraw-Hill Book Company New York, 1 v. (various pagings) :
- Pascale, R. (1991). "The Two Faces of Learning". In: *Modern Office Technology*, pp. 14–16.
- Payne, J.A. (1982). *Introduction to Simulation: Programming Techniques and Methods Analysis*. McGraw-Hill Computer Science Series. McGraw-Hill. ISBN: 9780070489455. URL: <https://books.google.de/books?id=9e1QAAAAMAAJ>.
- Pedler, M., J. Burgoyne, and T. Boydell (1991). *The Learning Company: A Strategy for Sustainable Development*. McGraw-Hill. ISBN: 9780077074791. URL: <https://books.google.de/books?id=QWHuAAAAMAAJ>.
- Peffers, Ken et al. (Aug. 2006). "The Design Science Research Process: A Model for Producing and Presenting Information Systems Research". In: *1st International Conference on Design Science in Information Systems and Technology (DESIST) 24.3*, pp. 83–106. URL: http://www.wrsc.org/sites/default/files/documents/000designsresearchproc_desist_2006.pdf.
- Peffers, Ken et al. (Aug. 2007). "A Design Science Research Methodology for Information Systems Research". In: *Management Informations Systems* 24.3, pp. 45–78. URL: http://wise.vub.ac.be/thesis_info/Design_Science_Research_Methodology_2008.pdf.

- Peinl, R. (2006). "A Knowledge Sharing Model illustrated with the Software Development Industry". In: *Proceedings of the Multikonferenz Wirtschaftsinformatik (MKWI 2006), Passau, Germany*. GIT Verlag.
- Peltu, Malcolm, Chris W Clegg, and Reg Sell (1996). *Business Process Engineering: The Human Issues*. ESRC Business Processes Resource Centre, University of Warwick.
- Pérez-Ortiz, J. A. et al. (2003). "Kalman filters improve LSTM network performance in problems unsolvable by traditional recurrent nets". In: *Neural Networks* 16.2, pp. 241–250.
- Peschl, M.F. (1990). *Cognitive Modelling: Ein Beitrag zur Cognitive Science aus der Perspektive des Konstruktivismus und des Konnektionismus*. DUV: Datenverarbeitung. Deutscher Universitätsverlag. URL: <https://books.google.de/books?id=Cjv3AAAACAAJ>.
- Peters, G. (1987). *Ablauforganisation und Informationstechnologie im Büro*. Köln.
- Peters, W. (1998). "Zur Theorie der Modellierung von Natur und Umwelt: Ein Ansatz zur Rekonstruktion und Systematisierung der Grundperspektiven ökologischer Modellbildung für planungsbezogene Anwendungen". PhD thesis. Berlin: Technische Universität Berlin.
- Petri, Carl Adam (1962). "Kommunikation mit Automaten". ger. PhD thesis. Universität Hamburg.
- Phan, Doantam et al. (2005). "Flow map layout". In: *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005*. Pp. 219–224. DOI: <https://doi.org/10.1109/INFVIS.2005.1532150>.
- Piehler, J. and H.U. Zschiesche (2013). *Simulationsmethoden*. Mathematik für Ingenieure und Naturwissenschaftler, Ökonomen und Landwirte. Vieweg+Teubner Verlag. ISBN: 9783322964243. URL: <https://books.google.de/books?id=1C6eBgAAQBAJ>.
- Pirkkalainen, Henri and Jan Pawłowski (2013). "Global social knowledge management: from barriers to the selection of social tools". In: *Electronic journal of knowledge management* 11.1.
- Plaut, D. C., S. J. Nowlan, and G. E. Hinton (1986). "Experiments on learning backpropagation". In: *Technical Report CMU-CS-86-126, Carnegie-Mellon University, Pittsburgh, PA*, p. 1.
- Poggio, T. and F. Girosi (1990). "Networks for approximation and learning". In: *Proceedings of the IEEE* 78.9, pp. 1481–1497. ISSN: 0018-9219. DOI: <https://doi.org/10.1109/5.58326>.
- Pogorzelska, Barbara (2009). *Arbeitsbericht (umfangreiche Beschreibung)—KMDLv2.2*. Tech. rep. University of Potsdam, Department of Business Informatics and Electronic Government.
- Pogorzelska, Barbara (2012). "KMDLv2.2: A semi-formal description language for modeling knowledge conversions". In: *Modeling and Analyzing knowledge intensive business processes with KMDL. Comprehensive insights into theory and practice*. Ed. by N. Gronau. GIT mbH Verlag Berlin.
- Pohl, H.C. (1977). *Problemorientierte Entscheidungsfindung in Organisationen*. Berlin: de Gruyter.
- Polanyi, M. and A. Sen (2009). *The Tacit Dimension*. University of Chicago Press. ISBN: 9780226672984. URL: <https://books.google.de/books?id=zfsb-eZHPy0C>.
- Poli, Riccardo, James Kennedy, and Tim Blackwell (2007). "Particle swarm optimization". In: *Swarm Intelligence* 1.1, pp. 33–57. ISSN: 1935-3820. DOI: <https://doi.org/10.1007/s11721-007-0002-0>. URL: <https://doi.org/10.1007/s11721-007-0002-0>.

- Popova-Zeugmann, Louchka (1991). "On Time Petri Nets". In: *Elektronische Informationsverarbeitung und Kybernetik* 27.4, pp. 227–244.
- Popp, W. (1970). "Die Funktion von Modellen in der didaktischen Theorie". In: *Unterrichtsforschung und didaktische Theorie*. Ed. by G. Dohmen, F. Maurer, and W. Popp. Erziehung in Wissenschaft und Praxis. Piper, pp. 49–60. ISBN: 9783492018135. URL: <https://books.google.de/books?id=asZvQgAACAAJ>.
- Pouille, Frédéric and Massimo Scanziani (2004). "Routing of spike series by dynamic circuits in the hippocampus." In: *Nature* 429 6993, pp. 717–23.
- Prechtl, P. and F.P. Burkard (2008). *Metzler Philosophie Lexikon: Begriffe und Definitionen*. 3rd ed. J.B. Metzler. DOI: <https://doi.org/10.1007/978-3-476-05469-2>.
- Prechtl, P. and F.P. Burkard (2015). *Metzler Lexikon Philosophie: Begriffe und Definitionen*. J.B. Metzler. ISBN: 9783476054692. DOI: <https://doi.org/10.1007/978-3-476-05469-2>. URL: <https://books.google.de/books?id=T-5FDwAAQBAJ>.
- Preskill, H. and R.T. Torres (1998). *Evaluative Inquiry for Learning in Organizations*. SAGE Publications. ISBN: 9780761904533. URL: <https://books.google.de/books?id=nQujtAEACAAJ>.
- Priesse, L. and H. Wimmel (2008). *Petri-Netze*. EXamen. press Series. Springer Berlin Heidelberg. ISBN: 9783540769712. URL: <https://books.google.de/books?id=b6epmAEACAAJ>.
- Principe, Jose C., Neil R. Euliano, and W. Curt Lefebvre (1999). *Neural and Adaptive Systems: Fundamentals Through Simulations with CD-ROM*. 1st. New York, NY, USA: John Wiley & Sons, Inc. ISBN: 0471351679.
- PRISM (1986). PRISM: *Dispersion and Interconnection: Approaches to Distributed Systems Architecture*. CSC Index. Cambridge, MA: Hammer & Company Inc.
- Probst, G., S. Raub, and K. Romhardt (1997). *Wissen managen—Wie Unternehmen ihre wertvollste Ressource optimal nutzen*. Frankfurt am Main: Frankfurter Allgemeine Zeitung für Deutschland.
- Probst, G., S. Raub, and K. Romhardt (2006). *Wissen managen—Wie Unternehmen ihre wertvollste Ressource optimal nutzen*. 5th ed. Gabler Verlag.
- Probst, G.J.B. (1994). *Organisationales Lernen: Wettbewerbsvorteil der Zukunft*. Gabler Verlag. URL: <https://books.google.de/books?id=LaWkAAACAAJ>.
- Prudent, Y. and A. Ennaji (2005). "An incremental growing neural gas learns topologies". In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks*, 2005. Vol. 2, 1211–1216 vol. 2. DOI: <https://doi.org/10.1109/IJCNN.2005.1556026>.
- Radhakrishnan, R. and S. Balasubramanian (2008). *Business Process Reengineering: Text and Cases*. PHI Learning. ISBN: 9788120335677. URL: <https://books.google.de/books?id=YSA7kb27a70C>.
- Rahner, K. and H. Vorgrimler (1969). *Internationale Dialog Zeitschrift*. Bd. 2. Verlag der Internationalen Dialog Zeitschrift. URL: <https://books.google.de/books?id=M0pZywquhx0C>.
- Ranjbarfard, Mina et al. (2013). "Identifying knowledge management problems using a process-based method (a case study of process 137)". In: *Business Proc. Manag. Journal* 19.2, pp. 263–291. DOI: <https://doi.org/10.1108/14637151311308312>. URL: <https://doi.org/10.1108/14637151311308312>.
- Ranvier, Louis (1872). "Recherches sur l'histologie et la physiologie des nerfs". In: *Arch Physiol Norm Pathol* 4, pp. 129–49.

- Ray, Arijit et al. (2019). "Lucid Explanations Help: Using a Human-AI Image-Guessing Game to Evaluate Machine Explanation Helpfulness". In: *arXiv preprint arXiv*: 1904.03285.
- Ray, Subhasis and Upinder Bhalla (2008). "PyMOOSE: interoperable scripting in Python for MOOSE". In: *Frontiers in Neuroinformatics* 2, p. 6. ISSN: 1662-5196. DOI: <https://doi.org/10.3389/neuro.11.006.2008>. URL: <https://www.frontiersin.org/article/10.3389/neuro.11.006.2008>.
- Reber, Arthur S. (1989). "Implicit learning and tacit knowledge". In: *Journal of Experimental Psychology: General*, p. 219235.
- Rechenberg, P. (2006). *Informatik-Handbuch*. Hanser. ISBN: 9783446401853. URL: <https://books.google.de/books?id=N4V2q941AD8C>.
- Reichelt, H. (2014). *Spiritualität und die Wissenschaft*. disserta Verlag. ISBN: 9783954255429. URL: <https://books.google.de/books?id=8CERBAAQBAJ>.
- Reihlen, M. (1997). "Ansätze in der Modelldiskussion: Eine Analyse der Passivistischen Abbildungsthese und der Aktivistischen Konstruktionsthese". In: *Arbeitsberichte des Seminars für Allgemeine Betriebswirtschaftslehre, Betriebswirtschaftliche Planung und Logistik der Universität zu Köln*. 92. Delfmann, W.
- Reinmann-Rothmeier, Gabi (2001). *Wissen managen: Das Münchener Modell*. URL: <http://nbn-resolving.de/urn/resolver.pl?urn=nbn:de:bvb:19-epub-239-4>.
- Reinmann-Rothmeier, Gabi and Heinz Mandl (2000). "Wissen". In: *Lexikon der Neurowissenschaft*. Spektrum Akademischer Verlag, Heidelberg.
- Remmen, Arne (2007). *Life cycle management: a business guide to sustainability*. UNEP/Earthprint.
- Remus, Ulrich (2002). *Prozessorientiertes Wissensmanagement. Konzepte und Modellierung*. URL: <http://epub.uni-regensburg.de/9925/>.
- Ren, Changrui, Yueling Chai, and Yi Liu (2005). "Adding Value to System Dynamics Modeling by Using Artificial Neural Network". In: *Advances in Neural Networks—ISNN 2005*. Ed. by Jun Wang, Xiao-Feng Liao, and Zhang Yi. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 430–435. ISBN: 978-3-540-320678.
- Renzl, Birgit (2004). "Zentrale Aspekte des Wissensbegriffs-Kernelemente der Organisation von Wissen". In: Wyssusek, B.(Hg.), *Wissensmanagement komplex*. Berlin, pp. 27–42.
- Rescher, N. (2000). *Process Philosophy: A Survey of Basic Issues*. University of Pittsburgh Press. URL: <https://books.google.de/books?id=E6rOyLVeBOcC>.
- Rescher, N (2006). *Process Philosophical Deliberations*. Process Thought. DeGruyter. ISBN: 9783110328325. URL: <https://books.google.de/books?id=rvxangM29i8C>.
- Rhodes, Carl (1998). "Chris Argyris and Donald A. Schön (1996): Organizational learning II: Theory, method and practice Reading, MA: Addison-Wesley, XXIX + 305 pp, \$41.95 ISBN 0-201-62983-6". In: *Asia Pacific Journal of Human Resources* 36.1, pp. 107–109. DOI: <https://doi.org/10.1177/103841119803600112>. eprint: <https://onlinelibrary.wiley.com/doi/abs/10.1177/103841119803600112>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1177/103841119803600112>.
- Ribeiro, S. and R. Alquézar (2002). "Incremental Construction of LSTM Recurrent Neural Network". In: *Proceedings of the seventh Iberoamerican Congress on Pattern Recognition* 7, pp. 171–184. URL: <http://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735?journalCode=neco>.
- Richardson, G. (1997). "Problems in causal loop diagrams revisited". In: *System Dynamics Review* 13.3, pp. 247–252.

- Richardson, Gary L., Brad M. Jackson, and Gary W. Dickson (Dec. 1990). "A Principles-based Enterprise Architecture: Lessons from Texaco and Star Enterprise". In: *MIS Q.* 14.4, pp. 385–403. ISSN: 0276-7783. DOI: <https://doi.org/10.2307/249787>. URL: <http://dx.doi.org/10.2307/249787>.
- Richardson, George P. (1986). "Problems with causal-loop diagrams". In: *System Dynamics Review* 2.2, pp. 158–170. DOI: <https://doi.org/10.1002/sdr.4260020207>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/sdr.4260020207>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/sdr.4260020207>.
- Richter, Alexander (2008). *IT-gestütztes Wissensmanagement: Theorie, Anwendungen und Barrieren*. BoD-Books on Demand.
- Riedmiller, M. and H. Braun (1993). "A direct adaptive method for faster backpropagation learning: The RPROP algorithm". In: *Proc. of the IEEE Intl. Conf. on Neural Networks, San Francisco, CA*, pp. 586–591. URL: citeseer.ist.psu.edu/riedmiller93irect.html.
- Riekhof, H.C. (1997). *Beschleunigung von Geschäftsprozessen. Wettbewerbsvorteile durch Lernfähigkeit*. Stuttgart: Schäffer-Poeschel.
- Rieper, B. (1992). *Betriebswirtschaftliche Entscheidungsmodelle: Grundlagen*. NWB-Betriebswirtschaft. Verlag Neue Wirtschafts-Briefe. ISBN: 9783482452512. URL: <https://books.google.de/books?id=bxMtPQAACAAJ>.
- Rigdon, W.B. (1989). "Architectures and Standards". In: *Information Management Directions: The Integration Challenge (NIST Special Publication 500–167)*. Ed. by E.N. Fong and A.H. Goldfine. Gaithersburg, MD: National Institute of Standards and Technology (NIST), pp. 135–150.
- Ritter, H., T. Martinetz, and K. Schulten (1991). *Neuronale Netze: eine Einführung in die Neuroinformatik selbstorganisierender Netzwerke*. Reihe künstliche Intelligenz. Addison-Wesley. ISBN: 9783893191314. URL: <https://books.google.de/books?id=MfsARQAACAAJ>.
- Robinson, A. J. and F. Fallside (1987). "The utility driven dynamic error propagation network". In: *Technical Report CUED/F-INFENG/TR.1, Cambridge University Engineering Department*, p. 1.
- Rohloff, Michael (1995). "Integrierte Informationssysteme durch Modellierung von Geschäftsprozessen". In: *Wirtschaftsinformatik '95*. Ed. by Wolfgang König. Heidelberg: Physica-Verlag HD, pp. 83–97. ISBN: 978-3-642-57873-1.
- Rolland, C., C. Souveyet, and C. B. Achour (1998). "Guiding goal modeling using scenarios". In: *IEEE Transactions on Software Engineering* 24.12, pp. 1055–1071.
- Rolland, Colette, Naveen Prakash, and Adolphe Benjamen (1999). "A multi-model view of process modelling". In: *Requirements engineering* 4.4, pp. 169–187.
- Rolls, E. and G. Deco (2001). *Computational Neuroscience of Vision*. OUP Oxford. ISBN: 9780198524885. URL: <https://books.google.de/books?id=SbFpuQAAACAAJ>.
- Rosemann, M. (1996). *Komplexitätsmanagement in Prozessmodellen: Methodenspezifische Gestaltungsempfehlungen für die Informationsmodellierung*. Schriften zur EDV-orientierten Betriebswirtschaft. Gabler Verlag. ISBN: 9783409121729. URL: <https://books.google.de/books?id=t9WvAAAACAAJ>.
- Rosemann, Michael, Ansgar Schwemann, and Patrick Delfmann (2012). "Vorbereitung der Prozessmodellierung". In: *Prozessmanagement: Ein Leitfaden zur prozessorientierten Organisationsgestaltung*. Ed. by Jörg Becker, Martin Kugeler, and Michael Rosemann. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 47–111. ISBN: 978-3-642-33844-1.

- DOI: https://doi.org/10.1007/978-3-642-33844-1_3. URL: https://doi.org/10.1007/978-3-642-33844-1_3.
- Rosenblatt, F. (1958). "The perceptron: A probabilistic model for information storage and organization in the brain". In: *Psychological Review* 65, pp. 386–408.
- Rosenblatt, F. (1963). "Principles of Neurodynamics". In: *Spartan, New York*, p. 1.
- Rosing, Mark von et al. (2015). "Business Process Model and Notation (BPMN)". In: *The Complete Business Process Handbook*. Ed. by Mark von Rosing, August-Wilhelm Scheer, and Henrik von Scheel. Boston: Morgan Kaufmann, pp. 433–457. ISBN: 978-0-12-799959-3. DOI: <https://doi.org/10.1016/B9780-12-799959-3.00021-5>. URL: <http://www.sciencedirect.com/science/article/pii/B9780127999593000215>.
- Rothman, Jason S. et al. (2009). "Synaptic depression enables neuronal gain control". In: *Nature* 457.7232, pp. 1015–8.
- Rougier, Nicolas (2019). *Biological Neuron Schema*. Tech. rep. CC-BY-SA-3.0. URL: <https://commons.wikimedia.org/w/index.php?curid=2192116>.
- Ruiz, Valentin Valero, David de Frutos-Escríg, and Fernando Cuartero (1991). "Simulation of Timed Petri Nets by Ordinary Petri Nets and Applications to Decidability of the Timed Reachability Problem and other Related Problems". In: *Proceedings of the Fourth International Workshop on Petri Nets and Performance Models, PNPM 1991, Melbourne, Victoria, Australia, December 2–5, 1991*, pp. 154–163. DOI: <https://doi.org/10.1109/PNPM.1991.238772>. URL: <https://doi.org/10.1109/PNPM.1991.238772>.
- Rumbaugh, James et al. (1991). *Object-oriented Modeling and Design*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc. ISBN: 0-13-629841-9.
- Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). "Learning internal representations by error propagation". In: *MIT Press, Cambridge, MA, USA, ISBN 0-262-68053-X*, pp. 318–362.
- Russell, C. Eberhart (2007). *Computational Intelligence: Concepts to Implementations*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. ISBN: 1558607595, 9780080553832.
- Russell, N. et al. (Jan. 2004). *Workflow Data Patterns (FIT-TR-2004-01, QUT Technical report)*. Tech. rep. Queens-land University of Technology, Brisbane.
- Russell, S. and P. Norvig (2003). *Artificial Intelligence: A Modern Approach*. Second. Series in Artificial Intelligence. Upper Saddle River, NJ: Prentice Hall.
- Russell, S. and P. Norvig (2010). *Artificial Intelligence: A Modern Approach*. Third. Series in Artificial Intelligence. Upper Saddle River, NJ: Prentice Hall. URL: <http://aima.cs.berkeley.edu/>.
- Ruszczynski, A. (2011). *Nonlinear Optimization*. Princeton University Press. ISBN: 9781400841059. URL: <https://books.google.de/books?id=41dhYmzMSm4C>.
- Salakhutdinov, Ruslan, Andriy Mnih, and Geoffrey Hinton (2007). "Restricted Boltzmann Machines for Collaborative Filtering". In: *Proceedings of the 24th International Conference on Machine Learning*. ICML '07. Corvalis, Oregon, USA: ACM, pp. 791–798. ISBN: 978-1-59593-793-3. DOI: <https://doi.org/10.1145/1273496.1273596>. URL: <http://doi.acm.org/10.1145/1273496.1273596>.
- Salzmann, C. (1974). "Die Bedeutung des Modellbegriffs in Unterrichtsforschung und Unterrichtsplanung". In: *Unterrichtsanalysen in der Diskussion*. Ed. by L. Roth, G. Petrat, and A. Ammen. Schroedel, pp. 171–205.

- Sanger, Terence D. (1989). "Optimal unsupervised learning in a single-layer linear feed-forward neural network". In: *Neural Networks* 2.6, pp. 459–473. ISSN: 0893-6080. DOI: [https://doi.org/10.1016/0893-6080\(89\)90044-0](https://doi.org/10.1016/0893-6080(89)90044-0). URL: <http://www.sciencedirect.com/science/article/pii/0893608089900440>.
- Sarma, Gopal P. et al. (2018). "OpenWorm: overview and recent advances in integrative biological simulation of <i>Caenorhabditis elegans</i>". In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 373.1758, p. 20170382. DOI: <https://doi.org/10.1098/rstb.2017.0382>, eprint: <https://royalsocietypublishing.org/doi/pdf/10.1098/rstb.2017.0382>. URL: <https://royalsocietypublishing.org/doi/abs/10.1098/rstb.2017.0382>.
- Schalkoff, R.J. (1997). *Artificial neural networks*. McGraw-Hill series in computer science: Artificial intelligence. McGraw-Hill. ISBN: 9780070571181. URL: <https://books.google.de/books?id=bbNQAAAAMAAJ>.
- Schaal, T. et al. (2010). "PyBrain". In: *Journal of Machine Learning Research* 11, pp. 743–746. URL: <https://github.com/pyBrain/pyBrain/releases>.
- Scheeben, M.J. (1875). *Periodische Blätter zur Wissenschaftlichen Besprechung der Religiösen Fragen der Gegenwart*. Bd. 4. URL: <https://books.google.de/books?id=J6IwAAAAYAAJ>.
- Scheer, A.-W. (2002). *ARIS—Vom Geschäftsprozess zum Anwendungssystem*. 4. Berlin: Springer.
- Scheer, A.W. (1969). *Die industrielle Investitionsentscheidung: Eine theoretische und empirische Untersuchung zum Investitionsverhalten in Industrieunternehmungen*. Betriebswirtschaftliche Forschung zur Unternehmensführung. Gabler Verlag. ISBN: 9783322983817. URL: <https://books.google.de/books?id=8q6TngEACAAJ>.
- Scheer, A.W. (1976). *Produktionsplanung auf der Grundlage einer Datenbank des Fertigungsbereichs*. 1. Oldenbourg Verlag München Wien.
- Scheer, A.W. (1997). *Wirtschaftsinformatik: Referenzmodelle für industrielle Geschäftsprozesse*. Springer Berlin Heidelberg. ISBN: 9783540629672. URL: <https://books.google.de/books?id=LKHTM-9dqF4C>.
- Scheer, A.W. (1998). *ARIS—Modellierungsmethoden, Metamodelle, Anwendungen*. 3. Springer Berlin Heidelberg. ISBN: 9783642977329. URL: <https://books.google.de/books?id=vwLGjwEACAAJ>.
- Scheer, A.W. (1999). *ARIS—Business Process Modeling*. ARIS—Business process modeling Bd. 2. Springer. URL: <https://books.google.de/books?id=h5srAQAAAMAAJ>.
- Scheer, A.W. et al. (2003). *Business Process Change Management: ARIS in Practice*. Springer. ISBN: 9783540002437. URL: <https://books.google.de/books?id=LrBf5pE8UdAC>.
- Schlabach, P. (2018). *Sitte, Ethik und Moral: eine Begründung*. tredition. ISBN: 9783746970080. URL: <https://books.google.de/books?id=wcRtDwAAQBAJ>.
- Schlagheck, B. (2000). "Objektorientierte Referenzmodelle für das Prozeß- und Projektcontrolling, Grundlagen, Konstruktion, Anwendungsmöglichkeiten". PhD thesis. Wiesbaden: Univ. Münster.
- Schmidhuber, J. (2013). "PyBrain as the State of the Art". In: *Internet Homepage*, Hauptfenster. URL: www.idsia.ch/~juergen/rnn.html.
- Schmidhuber, Jürgen (2015). "Deep learning in neural networks: An overview". In: *Neural Networks* 61, pp. 85–117. ISSN: 0893-6080. DOI: <http://dx.doi.org/10.1016/j.neunet.2014.09.003>. URL: <http://www.sciencedirect.com/science/article/pii/S0893608014002135>.

- Schmidt, C. (2000). "Arbeitsgedächtnis und fremdsprachliches Leseverständnis". In: *Zeitschrift für Fremdsprachenforschung* 1.11, pp. 83–101.
- Schmidt, Günther (1980). *Simulationstechnik*. Oldenbourg.
- Schmidt, J.W. and R.E. Taylor (1970). *Simulation and Analysis of Industrial Systems*. The Irwin series in quantitative analysis for business. Richard D. Irwin. URL: <https://books.google.de/books?id=UNZEAAAIAAJ>.
- Schmidt, R.F. et al. (2013). *Grundriß der Neurophysiologie*. Heidelberger Taschenbücher. Springer Berlin Heidelberg. ISBN: 9783642962301. URL: <https://books.google.de/books?id=L5eiBgAAQBAJ>.
- Schmitz, C. and B. Zucker (2003). *Wissensmanagement: schnelleres Lernen im Unternehmen*. Metropolitan professional. Metropolitan-Verlag. ISBN: 9783896233196. URL: <https://books.google.de/books?id=mQvAOwAACAAJ>.
- Schnabel, Ulrich G (2013). *Management des intellektuellen Kapitals wissensintensiver Dienstleister: Strategieoptionen zum Erwerb und zur Entwicklung intellektueller Ressourcen*. Springer-Verlag.
- Schneider, Ursula (1996). *Wissensmanagement: die Aktivierung des intellektuellen Kapitals*. Frankfurter Allgemeine Zeitung, Verlag-Bereich Wirtschaftsbücher.
- Schnell, R., P. B. Hill, and E. Esser (2011). *Methoden der empirischen Sozialforschung*. Oldenbourg Wissenschaftsverlag. ISBN: 9783486591064. URL: <https://books.google.de/books?id=d37SHMEuJAkC>.
- Scholz, Rainer (1994). *Geschäftsprozeßoptimierung: crossfunktionale Rationalisierung oder strukturelle Reorganisation*. deu. Dissertation u.a. Prüfungsschriften.
- Schön, Donald A. (1975). "Deutero-learning in organizations: Learning for". In: *Organizational Dynamics* 4.1, pp. 2–16. ISSN: 0090-2616. DOI: [https://doi.org/10.1016/0090-2616\(75\)90001-7](https://doi.org/10.1016/0090-2616(75)90001-7). URL: <http://www.sciencedirect.com/science/article/pii/0090261675900017>.
- Schulze, D. (2001). "Grundlagen der wissensbasierten Konstruktion von Modellen betrieblicher Systeme". PhD thesis. Aachen: Universität Bamberg.
- Schuster, Thomas (2012). "Modellierung, Integration und Analyse von Ressourcen in Geschäftsprozessen". German. PhD thesis. 286 pp. ISBN: 978-3-86644-889-6. DOI: <https://doi.org/10.5445/KSP/1000029027>.
- Schütte, R. (1998). *Grundsätze ordnungsmäßiger Referenzmodellierung: Konstruktion konfigurations- und anpassungsorientierter Modelle*. neue betriebswirtschaftliche forschung (nbf). Gabler Verlag. URL: <https://books.google.de/books?id=4a3IAAAACAAJ>.
- Schütte, R. (2013). *Grundsätze ordnungsmäßiger Referenzmodellierung: Konstruktion konfigurations- und anpassungsorientierter Modelle*. neue betriebswirtschaftliche forschung (nbf). Gabler Verlag. ISBN: 9783663102335. URL: <https://books.google.de/books?id=maKABwAAQBAJ>.
- Schwaiger, R. and J. Steinwendner (2019). *Neuronale Netze programmieren mit Python: Ihre Einführung in die Künstliche Intelligenz. Inkl. KI-Lernumgebung und Einstieg in TensorFlow*. Rheinwerk Computing. RheinwerkVerlagGmbH. ISBN: 9783836261425. URL: <https://books.google.de/books?id=KIDwuQEACAAJ>.
- Schwartz, P. and J.A Ogilvy (1998). "Scenarios for global investment strategy for the new century." In: ed. by L. Fahey and R. Randall. *Learning from the Future*. John Wiley and Sons, New York, pp. 175–186.

- Schweitzer, M. (1967). "Methodologische und entscheidungstheoretische Grundfragen der betriebswirtschaftlichen Prozeßstrukturierung". In: *Zeitschrift für betriebswirtschaftliche Forschung* 19, pp. 279–296.
- Schwenker, F. and C. Dietrich (2000). "Initialization of radial basis function networks by means of classification trees". In: *Neural Network World* 10, pp. 473–482.
- Schwenker, F et al. (1994). "Similarities of LVQ and RBF learning-a survey of learning rules and the application to the classification of signals from high-resolution electrocardiography". In: *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*. Vol. 1. IEEE, pp. 646–651.
- Schwenker, Friedhelm, Hans A. Kestler, and Günter Palm (2001). "Three learning phases for radial-basisfunction networks". In: *Neural Networks* 14.4, pp. 439–458. ISSN: 0893-6080. DOI: [https://doi.org/10.1016/S0893-6080\(01\)00027-2](https://doi.org/10.1016/S0893-6080(01)00027-2). URL: <http://www.sciencedirect.com/science/article/pii/S0893608001000272>.
- Schwermer, M. (1998). *Modellierungsvorgehen zur Planung von Geschäftsprozessen*. Berlin: IPK.
- Schwicket, A.C. and K. Fischer (1996). *Der Geschäftsprozeß als formaler Prozeß: Definition, Eigenschaften und Arten*. Arbeitspapiere WI. URL: <https://books.google.de/books?id=Zma4tgAACAAJ>.
- Schwicket, A.C. and L.F. Rey (1996). *Manuelle und elektronische Vorgangssteuerung*. Arbeitspapiere WI. URL: <https://books.google.de/books?id=kXultgAACAAJ>.
- Seibt, Johanna (2018). "Process Philosophy". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Spring 2018. Metaphysics Research Lab, Stanford University.
- Selvaraju, Ramprasaath R. et al. (2016). "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 618–626.
- Senge, P. (1990a). "The Leader's New Work: Building Learning Organizations". In: *Sloan Management Review*, pp. 7–23.
- Senge, Peter M. and John D. Sterman (1992). "Systems thinking and organizational learning: Acting locally and thinking globally in the organization of the future". In: *European Journal of Operational Research* 59.1, pp. 137–150. URL: <https://EconPapers.repec.org/RePEc:eee:ejores:v:59:y:1992:i:1:p:137-150>.
- Senge, P.M. (1990b). *The Fifth Discipline: The Art and Practice of the Learning Organization*. A Currency book. Doubleday/Currency. ISBN: 9780385260947. URL: <https://books.google.de/books?id=bVZqAAAMAAJ>.
- Senge, P.M. et al. (1994). *The Fifth Discipline Fieldbook: Strategies and Tools for Building a Learning Organization*. A currency book. Currency, Doubleday. ISBN: 9780385472562. URL: <https://books.google.de/books?id=thTrQVR8EIC>.
- Serenko, Alexander and Nick Bonis (2009). "Global ranking of knowledge management and intellectual capital academic journals". In: *Journal of Knowledge Management*.
- Sergeant, Joseph (2000). "The cognitive-energetic model: an empirical approach to attention-deficit hyperactivity disorder". In: *Neuroscience & Biobehavioral Reviews* 24.1, pp. 7–12.
- Sergeant, Joseph A (2005). "Modeling attention-deficit/hyperactivity disorder: a critical appraisal of the cognitive-energetic model". In: *Biological psychiatry* 57.11, pp. 1248–1255.

- Shabani, Mohsen Ostad and Ali Mazahery (2012). "Optimization of process conditions in casting aluminum matrix composites via interconnection of artificial neurons and progressive solutions". In: *Ceramics International* 38.6, pp. 4541–4547.
- Shannon, C.E. and W. Weaver (1949). *The Mathematical Theory of Communication*. The Mathematical Theory of Communication Bd. 1. University of Illinois Press. URL: <https://books.google.de/books?id=ZiYIAQAAIAAJ>.
- Shannon, Claude E. and Warren Weaver (1963). *A Mathematical Theory of Communication*. Champaign, IL, USA: University of Illinois Press. ISBN: 0252725484.
- Shannon, R.E. (1975). *Systems simulation: the art and science*. Prentice-Hall. ISBN: 9780138818395. URL: <https://books.google.de/books?id=JGpRAAAAMAAJ>.
- Sheffield, James (2009). "Pluralism in Knowledge Management: A Review." In: *Electronic Journal of Knowledge Management* 7.3.
- Shelly, G.B. and M.E. Vermaat (2011). *Discovering Computers, Complete: Your Interactive Guide to the Digital World*. MindTap Course List Series. Cengage Learning. ISBN: 9781111530327. URL: https://books.google.de/books?id=icZMX_fHHNgC.
- Shen, Changyu, Lixia Wang, and Qian Li (2007). "Optimization of injection molding process parameters using combination of artificial neural network and genetic algorithm method". In: *Journal of Materials Processing Technology* 183.2–3, pp. 412–418.
- Shepard, Roger N and Phipps Arabie (1979). "Additive clustering: Representation of similarities as combinations of discrete overlapping properties." In: *Psychological Review* 86.2, p. 87.
- Shewchuk, J. R. (1994). "An introduction to the conjugate gradient method without the agonizing pain". In: *Technical Report, Carnegie Mellon university, Pittsburgh, PA, USA*, p. 1.
- Shewhart, W. A. and W. E. Deming (1939). *Statistical method from the view-point of quality control*. English. IX + 155 p. Washington, D.C., The Graduate School, Department of Agriculture (1939).
- Shishvan, Masoud Soleymani and Jörg Benndorf (2017). "Operational Decision Support for Material Management in Continuous Mining Systems: From Simulation Concept to Practical Full-Scale Implementations". In: *Minerals* 7.7. ISSN: 2075-163X. DOI: <https://doi.org/10.3390/min7070116>. URL: <http://www.mdpi.com/2075-163X/7/7/116>.
- Shrivastava, Paul (1983). "A Typology of Organizational Learning Systems". In: *Journal of Management Studies* 20.1, pp. 7–28. DOI: <https://doi.org/10.1111/j.1467-6486.1983.tb00195.x>.
- Shrivastava, Paul (1985). "Knowledge systems for strategic decision making". In: *The Journal of Applied Behavioral Science* 21.1, pp. 95–107.
- Signavio GmbH (Jan. 2019). *Signavio Webpage*. URL: <https://www.signavio.com/> (visited on 01/09/2019).
- Signavio Team (2015). *Signavio Process Manager*. Practitioner Paper 1. Signavio Inc. URL: <https://www.signavio.com/wp-content/uploads/2015/06/Signavio-Process-Manager-EN-2017WEB.pdf> (visited on 01/09/2019).
- Signavio Team (2017). *Signavio Products Feature Overview*. Practitioner Paper 1. Signavio Inc. URL: https://www.signavio.com/wp-content/uploads/2017/11/Signavio_Features_en.pdf (visited on 01/09/2019).
- Sikström, Sverker and Göran Söderlund (2007). "Stimulus-dependent dopamine release in attention-deficit hyperactivity disorder". In: *Psychological review* 114.4, p. 1047.

- Simon, Daniel, Kai Fischbach, and Detlef Schoder (2013). "An Exploration of Enterprise Architecture Research". In: *Communications of the Association for Information Systems* 32.1.
- Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman (2014). "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14–16, 2014, Workshop Track Proceedings*. URL: <http://arxiv.org/abs/1312.6034>.
- Simpson, S. (2004). *An XML Representation for Crew Procedures*. Final Report. NASA Faculty Fellowship Program (Johnson Space Center).
- Sinur, Jim and Janelle B. Hill (2010). *Magic Quadrant for Intelligent Business Process Management Suite*. Tech. rep. G00205212. Gartner Inc.
- Sinur, Jim et al. (2012). *Magic Quadrant for Intelligent Business Process Management Suite*. Tech. rep. G00224913. Gartner Inc.
- Skwara, G. (2016). *Mentale Kommunikation: Magisches Quadrat*. Books on Demand. ISBN: 9783743121409. URL: <https://books.google.de/books?id=EqlyDQAAQBAJ>.
- Skwara, G. (2018). *Spirituelle Rückführung: Eine Anleitung zur möglichen eigenständigen Durchführung*. Books on Demand. ISBN: 9783752830651. URL: <https://books.google.de/books?id=38hWDwAAQBAJ>.
- Learning, Belief, & Action in Organizational Work Groups—A Conceptual Model of Work Group Learning* (1993). Atlanta, GA: Academy of Management Meeting.
- Sloan, Robert H. and Ugo Buy (1996). "Reduction rules for time Petri nets". In: *Acta Informatica* 33.7, pp. 687–706. ISSN: 1432-0525. DOI: <https://doi.org/10.1007/s002360050066>. URL: <https://doi.org/10.1007/s002360050066>.
- Smilkov, Daniel et al. (2017). "Smoothgrad: removing noise by adding noise". In: *arXiv preprint arXiv: 1706.03825*.
- Smith, Elizabeth A. (2001). "The role of tacit and explicit knowledge in the workplace". In: *Journal of Knowledge Management* 5.4, pp. 311–321. DOI: <https://doi.org/10.1108/13673270110411733>. eprint: <https://doi.org/10.1108/13673270110411733>. URL: <https://doi.org/10.1108/13673270110411733>.
- Snell, R.S. (2010). *Clinical Neuroanatomy*. Clinical Neuroanatomy. Wolters Kluwer Health/Lippincott Williams & Wilkins. ISBN: 9780781794275. URL: <https://books.google.de/books?id=ABPmvroyrD0C>.
- Solso, Robert L. (2005). *Kognitive Psychologie. mit 14 Tabellen*. ger. Springer-Lehrbuch. Heidelberg: Springer, VIII, 538 S. ISBN: 3-540-21270-1 and 978-3-540-21270-6.
- Sommerville, I. (1985). *Software Engineering (2Nd Ed.)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0-201-14229-5.
- Song, Sen et al. (Mar. 2005). "Highly Nonrandom Features of Synaptic Connectivity in Local Cortical Circuits". In: *PLOS Biology* 3.3. DOI: <https://doi.org/10.1371/journal.pbio.0030068>. URL: <https://doi.org/10.1371/journal.pbio.0030068>.
- Sowa, J.F. and J.A. Zachman (1992). "Extending and formalizing the framework for information systems architecture". In: *IBM Systems Journal* 31.3, pp. 590–616.
- Spaniol, Otto and Simon Hoff (1995). *Ereignisorientierte Simulation: Konzepte und Systemrealisierung*. Internat. Thomson Publ.
- Specker, A. (2005). *Modellierung von Informationssystemen: Ein methodischer Leitfaden zur Projektabwicklung*. vdf Wirtschaftsinformatik. vdf, Hochsch.-Verlag an der ETH. ISBN: 9783728129840. URL: <https://books.google.de/books?id=to2G1pHPNeUC>.

- Spies, P.D.M. and J. Trojan (2007). *Strategien zur Bewahrung von Wissen: Zur Sicherung nachhaltiger Wettbewerbsvorteile*. Deutscher Universitätsverlag. ISBN: 9783835090217. URL: <https://books.google.de/books?id=E4m4BSnICuAC>.
- Sporns, Olaf and Rolf Kotter (Oct. 2004). "Motifs in Brain Networks". In: *PLOS Biology* 2.11. DOI: <https://doi.org/10.1371/journal.pbio.0020369>. URL: <https://doi.org/10.1371/journal.pbio.0020369>.
- Springenberg, Jost Tobias et al. (2015a). "Striving for Simplicity: The All Convolutional Net". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Workshop Track Proceedings*. URL: <http://arxiv.org/abs/1412.6806>.
- Springenberg, J.T. et al. (2015b). "Striving for Simplicity: The All Convolutional Net". In: *ICLR (workshop track)*. URL: <http://lmb.informatik.uni-freiburg.de/Publications/2015/DB15a>.
- Spur, G., K. Mertins, and R Jochem (1993). *Integrierte Unternehmensmodellierung*. Berlin, Wien, Zürich: Beuth Verlag.
- Stachowiak, H. (1973). *Allgemeine Modelltheorie*. Springer-Verlag. ISBN: 9783211811061. URL: <https://books.google.de/books?id=DK-EAAAIAAJ>.
- Stachowiak, H. (1983). *Modelle—Konstruktion der Wirklichkeit*. Kritische Information. Fink. URL: <https://books.google.de/books?id=qfc3AQAAIAAJ>.
- Stamer, Dirk, Ole Zimmermann, and Kurt Sandkuhl (2016). "What Is a Framework?—A Systematic Literature Review in the Field of Information Systems". In: *Perspectives in Business Informatics Research*: vol. 261. Lecture Notes in Business Information Processing 261, pp. 145–158. ISBN: 978-3-319-45321-7.
- Sterman, J. (2000). *Business Dynamics: Systems Thinking and Modeling for a Complex World*. McGraw-Hill higher education. McGraw-Hill. ISBN: 9780071241076. URL: <https://books.google.de/books?id=hxxIPwAACAAJ>.
- Storkey, A.J. (1997). "Increasing the Capacity of the Hopfield Network without Sacrificing Functionality". In: *Lecture Notes on Computer Science 1327 (ICANN97)*. Springer Verlag, pp. 451–456.
- Strutz, Tilo (2016). *Data Fitting and Uncertainty: A Practical Introduction to Weighted Least Squares and Beyond*. Germany: Springer Vieweg. ISBN: 3834810223, 9783834810229.
- Stych, Christof and Klaus Zeppenfeld (2008). *ITIL*. Informatik im Fokus. Berlin: Springer. ISBN: 978-3-54073118-4. DOI: <https://doi.org/10.1007/978-3-540-73119-1>.
- Sulistio, Anthony, Chee Shin Yeo, and Rajkumar Buyya (2004). "A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools". In: *Software: Practice and Experience* 34.7, pp. 653–673. DOI: <https://doi.org/10.1002/spe.585>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.585>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.585>.
- Sullivan, C.H. (1985). "Systems Planning in the Information Age". In: *Sloan Management Review* 26.2, pp. 3–12.
- Sultanow, E. et al. (2012). "Modeling of Processes, Systems and Knowledge: a Multi-Dimensional Comparison of 13 Chosen Methods". In: *International Review on Computers and Software (IRECOS)* 6, pp. 3309–3319.
- Sun, R. and C. L. Giles (2001). "Sequence learning: from recognition and prediction to sequential decision making". In: *IEEE Intelligent Systems* 16.4, pp. 67–70. ISSN: 1541-1672. DOI: <https://doi.org/10.1109/MIS.2001.1463065>.

- Sundararajan, Mukund, Ankur Taly, and Qiqi Yan (2017). "Axiomatic Attribution for Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, pp. 3319–3328. URL: <http://proceedings.mlr.press/v70/sundararajan17a.html>.
- Süssenguth, W. (1991). *Methoden zur Planung und Einführung rechnergeführter Produktionsprozesse*. München: Hanser-Verlag.
- Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). "Sequence to Sequence Learning with Neural Networks". In: *CoRR* abs/1409.3215. arXiv: 1409.3215. URL: <http://arxiv.org/abs/1409.3215>.
- Sutton, Richard S. and Andrew G. Barto (1998). *Introduction to Reinforcement Learning*. 1st. Cambridge, MA, USA: MIT Press. ISBN: 0262193981.
- Sweller, John, Jeroen J. G. van Merriënboer, and Fred G. W. C. Paas (1998). "Cognitive Architecture and Instructional Design". In: *Educational Psychology Review* 10.3, pp. 251–296. ISSN: 1573-336X. DOI: <https://doi.org/10.1023/A:1022193728205>. URL: <https://doi.org/10.1023/A:1022193728205>.
- Szegedy, Christian et al. (2014). "Going Deeper with Convolutions". In: *CoRR* abs/1409.4842. arXiv: 1409.4842. URL: <http://arxiv.org/abs/1409.4842>.
- Szigeti, B et al. (2014). "OpenWorm: an open-science approach to modeling *Caenorhabditis elegans*". In: *Front Comput Neurosci* 8.137.
- Szyperski, N. and U. Winand (1974). *Entscheidungstheorie: eine Einführung unter besonderer Berücksichtigung spieltheoretischer Konzepte*. Betriebswirtschaftliche Studienbücher. Poeschel. ISBN: 9783791090832. URL: <https://books.google.de/books?id=kz82AAAACAAJ>.
- Tadel, François et al. (Jan. 2011). "Brainstorm: A User-friendly Application for MEG/EEG Analysis". In: *Intell. Neuroscience* 2011, 8:1–8:13. ISSN: 1687-5265. DOI: <https://doi.org/10.1155/2011/879716>. URL: <http://dx.doi.org/10.1155/2011/879716>.
- TAFIM (1996a). *Technical Architecture Framework for Information Management—DoD Standards-Based Architecture Planning Guide*. Vol. 4. 3. Arlington Country, VA: Defense Information Systems Agency, Defense Technical Information Center. URL: <https://books.google.de/books?id=YrGxDAEACAAJ>.
- TAFIM (1996b). *Technical Architecture Framework for Information Management—Overview*. Vol. 1. 3. Arlington Country, VA: Defense Information Systems Agency, Defense Technical Information Center.
- Tao, Fei et al. (2018). "Digital twin-driven product design, manufacturing and service with big data". In: *The International Journal of Advanced Manufacturing Technology* 94.9, pp. 3563–3576. ISSN: 1433-3015. DOI: <https://doi.org/10.1007/s00170-017-0233-1>. URL: <https://doi.org/10.1007/s00170-017-0233-1>.
- Tarski, A. (1935). *Der Wahrheitsbegriff in den formalisierten Sprachen*. Bd. 146. Studia philosophica. URL: <https://books.google.de/books?id=-xFbQAAACAAJ>.
- Tarski, Alfred (1944). "The Semantic Conception of Truth: And the Foundations of Semantics". In: *Philosophy and Phenomenological Research* 4.3, pp. 341–376.
- Tarski, A. (1954). "Contributions to the Theory of Models. I". In: *Indagationes Mathematicae (Proceedings)* 57, pp. 572–581. ISSN: 1385-7258. DOI: [https://doi.org/10.1016/S1385-7258\(54\)50074-0](https://doi.org/10.1016/S1385-7258(54)50074-0). URL: <http://www.sciencedirect.com/science/article/pii/S138572585400740>.

- Team SimPy (Jan. 2019a). *SimPy Repository*. URL: <https://bitbucket.org/simpy/simpy/> (visited on 01/06/2019).
- Team SimPy (Jan. 2019b). *SimPy Webpage*. URL: <https://simpy.readthedocs.io/en/latest/> (visited on 01/06/2019).
- Telesko, R., D. Karagiannis, and R. Woitsch (2001). "Wissensmanagement Systeme-Anwendungen-Technologien". In: ed. by N. Gronau. Aachen: Shaker Verlag Aachen. Chap. Knowledge Management Tools: The Promote Project, pp. 35–52.
- Teran, Conrad K and Delbert Grotewold (1996). *Process optimization and control system that plots inter-relationships between variables to meet an objective*. US Patent 5,521,814.
- Tergan, S.-O. (2004). "Psychologie des Wissensmanagements. Perspektiven, Theorien und Methoden." In: ed. by Gabi Reinmann and Heinz Mandl. Göttingen: Verl. f. Psychologie Hogrefe. Chap. Wissensmanagement mit Concept Maps, pp. 259–266. ISBN: 3-8017-1815-8. URL: <http://www.ciando.com/ebook/bid-4222>.
- Thalheim, Bernhard (Nov. 1, 2010). "Towards a Theory of Conceptual Modelling". In: *Journal of Universal Computer Science* 16.20, pp. 3102–3137.
- Thalheim, Bernhard (2012). "Transactions on Large-Scale Data-and Knowledge-Centered Systems VI". In: ed. by Abdelkader Hameurlain et al. Berlin, Heidelberg: Springer-Verlag. Chap. The Science and Art of Conceptual Modelling, pp. 76–105. ISBN: 978-3-642-34178-6. URL: <http://dl.acm.org/citation.cfm?id=2407076.2407079>.
- The AnyLogic Company (Nov. 2019). *AnyLogic Webpage*. URL: <https://www.anylogic.com/> (visited on 01/03/2019).
- The ARIS Community (Jan. 2019). *ARIS Community*. URL: <https://www.ariscommunity.com/> (visited on 01/16/2019).
- The BOC Group (Jan. 2019). *Adonis Webpage*. URL: <https://de.boc-group.com/adonis/> (visited on 01/19/2019).
- The Financial Times (1994). "Giant with Feed of Clay: Tom Lloyd Offers a Contrasting View of Business Process Reengineering". In: *Financial Times* 12, p. 8.
- The Scheer Group (Jan. 2019). *ARIS Webpage*. URL: <https://www.scheer-group.com/productssolutions/arис/> (visited on 01/16/2019).
- The Theano Development Team et al. (May 2016). "Theano: A Python framework for fast computation of mathematical expressions". In: *ArXiv e-prints*. arXiv: 1605.02688 [cs.SC].
- TheOpenGroup (2011). *TOGAF® Version 9.1*. TOGAF Series. van Haren Publishing. ISBN: 9789087536794. URL: <https://books.google.de/books?id=5-hEBAQBAJ>.
- Thim, Christof (2017). "Technologieakzeptanz in Organisationen". PhD thesis. University of Potsdam.
- Thomas, Oliver (2005). *Das Modellverständnis in der Wirtschaftsinformatik: Historie, Literaturanalyse und Begriffsexplikation*. Vol. 1. 184. Institut für Wirtschaftsinformatik (IWi) im DFKI, pp. 1–41.
- Thommen, Jean-Paul and Edeltraud Günther (2018). "Organisationales Lernen". In: *Springer Fachmedien Wiesbaden GmbH*. URL: <https://wirtschaftslexikon.gabler.de/definition/organisationaleslernen-44058/version-267379>.
- Thompson, James D. and William J. McEwen (1958). "Organizational Goals and Environment: Goal-Setting as an Interaction Process". In: *American Sociological Review* 23.1, pp. 23–31. ISSN: 00031224. URL: <http://www.jstor.org/stable/2088620>.
- Tjahjono, Benny and Xu Jiang (2015). "Linking Symbiotic Simulation to Enterprise Systems: Framework and Applications". In: *Proceedings of the 2015 Winter Simulation Conference*.

- WSC '15. Huntington Beach, California: IEEE Press, pp. 823–834. ISBN: 978-1-4673-9741-4. URL: <http://dl.acm.org/citation.cfm?id=2888619.2888714>.
- Toutenburg, H. et al. (2008). *Six Sigma: Methoden und Statistik für die Praxis*. Springer Berlin Heidelberg. ISBN: 9783540851370. URL: <https://books.google.de/books?id=5R2qciUSqSUC>.
- Trowitzsch, I. (2011). "Utilizing LSTM to Evaluate Rodent Brain Activity Measured by BOLD Time Series". MA thesis. Technical University Berlin.
- Tryon, R.C. (1939). *Cluster Analysis: Correlation Profile and Orthometric (factor) Analysis for the Isolation of Unities in Mind and Personality*. Edwards brother, Incorporated, lithoprinters and publishers. URL: <https://books.google.de/books?id=gsnrAAAAMAAJ>.
- Tu, Jack V. (1996). "Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes". In: *Journal of Clinical Epidemiology* 49.11, pp. 1225–1231. ISSN: 0895-4356. DOI: [http://dx.doi.org/10.1016/S0895-4356\(96\)00002-9](http://dx.doi.org/10.1016/S0895-4356(96)00002-9). URL: <http://www.sciencedirect.com/science/article/pii/S0895435696000029>.
- Tumay, Kerim (1995). "Business Process Simulation". In: *Winter Simulation Conference*.
- UK, itSMF (2012). *ITIL Foundation Handbook*. 3rd. Norwich: The Stationery Office.
- Ullrich, André et al. (2017). "Does Size Matter? The effect of enterprise size on the perception of benefits and risks of Open Innovation projects". In: *Journal of Innovation Management*.
- Ulrich, Frank (2010). *Zur methodischen Fundierung der Forschung in der Wirtschaftsinformatik*. Hubert Österle, Robert Winter, and Walter Brenner In: Gestaltungsorientierte Wirtschaftsinformatik: Ein Plädoyer für Rigor und Relevanz.
- University of Hamburg (Feb. 2019). *Petri Net Bibliography Webpage*. URL: <http://www.informatik.uni-hamburg.de/TGI/PetriNets/bibliographies/> (visited on 02/02/2019).
- VaezMousavi, S. Mohammad et al. (2007). "Evidence for differentiation of arousal and activation in normal adults". In: *Acta neurobiologiae experimentalis* 67.2, pp. 179–186.
- VDI, VDI-Fachbereich Fabrikplanung und -betrieb (1993). *Simulation von Logistik-, Materialfluss- und Produktionssystemen—Grundlagen*. Tech. rep. VDI-Gesellschaft Produktion und Logistik.
- Vella, Michael et al. (2014). "libNeuroML and PyLEMS: using Python to combine procedural and declarative modeling approaches in computational neuroscience". In: *Frontiers in Neuroinformatics* 8, p. 38. ISSN: 1662-5196. DOI: <https://doi.org/10.3389/fninf.2014.00038>. URL: <https://www.frontiersin.org/article/10.3389/fninf.2014.00038>.
- Venn M.A., J. (1880). "I. On the diagrammatic and mechanical representation of propositions and reasonings". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 10.59, pp. 1–18. DOI: <https://doi.org/10.1080/14786448008626877>. eprint: <https://doi.org/10.1080/14786448008626877>. URL: <https://doi.org/10.1080/14786448008626877>.
- Vernadat, F.B. (1996). *Enterprise Modeling and Integration: Principles and Applications*, London, UK: Chapman and Hall.
- Vernadat, François (1998). "The CIMOSA Languages". In: *Handbook on Architectures of Information Systems*. Ed. by Peter Bernus, Kai Mertins, and Günter Schmidt. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 243–263. ISBN: 978-3-662-03526-9. DOI: https://doi.org/10.1007/978-3-662-03526-9_11. URL: https://doi.org/10.1007/978-3-662-03526-9_11.

- Vidgen, Richard T. et al. (1994). "Business process reengineering: the need for a methodology to re-vision the organization". In: *Business Process Re-Engineering*. Vol. A-54. IFIP Transactions. Elsevier, pp. 603–612.
- Vikhar, P. A. (2016). "Evolutionary algorithms: A critical review and its future prospects". In: *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, pp. 261–265. DOI: <https://doi.org/10.1109/ICGTSPICC.2016.7955308>.
- Vinyals, Oriol et al. (2014). "Show and Tell: A Neural Image Caption Generator". In: *CoRR* abs/1411.4555. arXiv: 1411.4555. URL: <http://arxiv.org/abs/1411.4555>.
- Visser, Max (2003). "Gregory Bateson on deutero-learning and double bind: A brief conceptual history". In: *Journal of the History of the Behavioral Sciences* 39.3, pp. 269–278. DOI: <https://doi.org/10.1002/jhbs.10112>.
- Visser, Max (2007). "Deutero-Learning in Organizations: A Review and a Reformulation". In: *Academy of Management Review* 32.2, pp. 659–667. DOI: <https://doi.org/10.5465/amr.2007.24351883>.
- Von Krogh, P.M.D.I.M.G. et al. (2000). *Enabling Knowledge Creation: How to Unlock the Mystery of Tacit Knowledge and Release the Power of Innovation*. Oxford University Press, USA. ISBN: 9780195126167. URL: <https://books.google.de/books?id=5ZkGF6wtG2kC>.
- Vuorimaa, Petri (Sept. 1994). "Fuzzy Self-organizing Map". In: *Fuzzy Sets Syst.* 66.2, pp. 223–231. ISSN: 01650114. DOI: [https://doi.org/10.1016/0165-0114\(94\)90312-3](https://doi.org/10.1016/0165-0114(94)90312-3). URL: [http://dx.doi.org/10.1016/01650114\(94\)90312-3](http://dx.doi.org/10.1016/01650114(94)90312-3).
- Waegeman, Willem, Krzysztof Dembczyński, and Eyke Hüllermeier (2019). "Multi-target prediction: a unifying view on problems and methods". In: *Data Mining and Knowledge Discovery* 33.2, pp. 293–324.
- Walsh, James P. and Gerardo Rivera Ungson (1991). "Organizational Memory". In: *Academy of Management Review* 16.1, pp. 57–91. <https://doi.org/10.5465/amr.1991.4278992>.
- Wang, Zhijie et al. (2020). "A novel parallel clock-driven algorithm for simulation of neuronal networks based on virtual synapse". In: *SIMULATION* 96.4, pp. 415–427.
- Wardle, Carloine (1984). "The Evolution of Information Systems Architecture". In: *Proceedings of the 5th International Conference on Information Systems*. Ed. by L. Maggi, J.L. King, and K.L. Kraemer. Association for Information Systems. Tucson, AZ.
- Warnecke, G., A. Gissler, and G. Stammwitz (1998). "Referenzmodell Wissensmanagement—Ein Ansatz zur modellbasierten Gestaltung wissensorientierter Prozesse". In: *IM Information Management* 1, pp. 24–29.
- Warren, K. (2002). *Competitive Strategy Dynamics*. Wiley. ISBN: 9780471899495. URL: https://books.google.de/books?id=XH_XRFTsdwMC.
- Warta, Alexander (2011). *Kollaboratives Wissensmanagement in Unternehmen—Indikatoren für Erfolg und Akzeptanz am Beispiel von Wikis*. Boizenburg: Verlag Werner Hülsbusch.
- Warth, C.P. (2012). *Wissenstransferprozesse in der Automobilindustrie: Entwicklung eines ganzheitlichen Modells auf der Grundlage einer Praxisfallstudie*. Springer Link: Bücher. Gabler Verlag. ISBN: 9783834936578. URL: <https://books.google.de/books?id=8IUeBAAAQBAJ>.
- Watzlawick, P. (1984). *Die erfundene Wirklichkeit: Wie wissen wir, was wir zu wissen glauben?* 2. München: Piper.
- Weber, E. (2015). "Erarbeitung einer Methodik der Wandlungsfähigkeit". dissertation. Universität Potsdam, GIT mbH Verlag Berlin.

- Webster, Jane and Richard T. Watson (2002). "Analyzing the Past to Prepare for the Future: Writing a Literature Review". In: *MIS Quarterly* 26.2, pp. xiii–xxii. ISSN: 02767783. DOI: <https://doi.org/10.1.1.104.6570>. arXiv: 02767783.
- Wegener, Ingo (2005). "Simulated Annealing Beats Metropolis in Combinatorial Optimization". In: *Automata, Languages and Programming*. Ed. by Luís Caires et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 589–601. ISBN: 978-3-540-31691-6.
- Wei, Donglai et al. (2015). "Understanding Intra-Class Knowledge Inside CNN". In: *CoRR* abs/1507.02379. arXiv: 1507.02379. URL: <http://arxiv.org/abs/1507.02379>.
- Weick, Karl E. (1991). "The Nontraditional Quality of Organizational Learning". In: *Organization Science* 2.1, pp. 116–124. ISSN: 10477039, 15265455. URL: <http://www.jstor.org/stable/2634942>.
- Weinert, Franz Emanuel (2001). "Vergleichende Leistungsmessung in Schulen—eine umstrittene Selbstverständlichkeit". deu. In: *Leistungsmessungen in Schulen*. Ed. by Franz Emanuel Weinert. Weinheim: Beltz, pp. 17–32.
- Wellenreuther, Martin (2012). *Forschungsbasierte Schulpädagogik*. Schneider Verlag Hohengehren.
- Wendt, Wolf Rainer (1998). *Soziales Wissensmanagement*. Nomos-Verlag-Ges.
- Wenturis, Nikolaus, Walter Van Hove, and Volker Dreier (1992). *Methodologie der Sozialwissenschaft*. Tübingen: Francke Verlag.
- Wenzel, Wolfgang and Kay Hamacher (1999). "Stochastic tunneling approach for global minimization of complex potential energy landscapes". In: *Physical Review Letters* 82.15, p. 3003.
- Werbos, P. J. (1988). "Generalization of backpropagation with application to a recurrent gas market model". In: *Neural Networks*, p. 1.
- Werbos, P. J. (1990). "Backpropagation through time: What it does and how to do it". In: *Proceedings of the IEEE* 78.19, pp. 1550–1560.
- Westkämper, E. and T. Schmidt (1998). "Computer-assisted manufacturing process optimization with neural networks". In: *Journal of Intelligent Manufacturing* 9.4, pp. 289–294.
- White, S. (2004). *Business Process Modeling Notation (BPMN) Version 1.0*. Tech. rep. BPMI.org. URL: <http://www.bpmn.org/Documents/BPMN%20V1-0%20May%20202004.pdf>.
- Whitehead, Alfred North (1920). *The Concept of Nature: The Tarner Lectures Delivered in Trinity College, November 1919*. Dover Publications.
- Whitehead, Alfred North (1925). *Science and the Modern World*. New York: Free Press.
- Whitehead, Alfred North (1929). *Process and Reality: An Essay in Cosmology*. Free Press.
- Whitman, Larry, Brian L. Huff, and Adrien Presley (1998). "The Needs and Issues Associated with Representing and Integrating Multiple Views of the Enterprise". In: *Information Infrastructure Systems for Manufacturing II, IFIP TC5 WG5.3/5.7 Third International Working Conference on the Design of Information Infrastructure Systems for Manufacturing (DIISM '98), May 18–20, 1998, Fort Worth, Texas, USA*, pp. 139–152.
- Wiener, N. (1948). *Cybernetics; or, Control and communication in the animal and the machine*. Actualités scientifiques et industrielles. J. Wiley.
- Wiig, K.M. (2004). *People-focused Knowledge Management: How Effective Decision Making Leads to Corporate Success*. Elsevier Butterworth Heinemann. ISBN: 9780750677776. URL: <https://books.google.de/books?id=rcW2XUoZKwUC>.

- Wilde, Thomas and Thomas Hess (2006). *Methodenspektrum der Wirtschaftsinformatik: Überblick und Portfoliobildung*. Tech. rep. 2. München: Institut für Wirtschaftsinformatik und Neue Medien der Ludwig-Maximilians-Universität.
- Wildner, S. (2011). *Problemorientiertes Wissensmanagement: eine Neukonzeption des Wissensmanagements aus konstruktivistischer Sicht*. Reihe: Wirtschaftsinformatik. Eul Verlag. ISBN: 9783844100433.
- Williams, R. J. and D. Zipser (1995). "Gradient-based learning algorithms for recurrent networks and their computational complexity. In Y. Chauvin and D.E. Rumelhart, editors, Lawrence Erlbaum Publishers, Hillsdale, N.J." In: *Backpropagation: Theory, Architectures and Applications*, pp. 433–486. URL: cseer.nj.nec.com/williams95gradientbased.html.
- Willke, H. (2007). *Einführung in das systemische Wissensmanagement*. Carl-Auer compact. Auer. URL: <https://books.google.de/books?id=ACM8cAAACAAJ>.
- Wilson, Matthew A. and James M. Bower (1989). "Methods in Neuronal Modeling". In: ed. by Christof Koch and Idan Segev. Cambridge, MA, USA: MIT Press. Chap. The Simulation of Large-scale Neural Networks, pp. 291–333. ISBN: 0-262-11133-0. URL: <http://dl.acm.org/citation.cfm?id=94605.94623>.
- Wimmel, Harro (2008). *Entscheidbarkeit bei Petri Netzen: Überblick und Kompendium*. Springer, Berlin, Heidelberg. URL: <https://doi.org/10.1007/978-3-540-85471-5>.
- Wirth, Niklaus (Jan. 1983). "Program Development by Stepwise Refinement". In: *Commun. ACM* 26.1, pp. 70–74. ISSN: 0001-0782. DOI: <https://doi.org/10.1145/357980.358010>. URL: <http://doi.acm.org/10.1145/357980.358010>.
- Wirth, Niklaus (1986). *Algorithms & Data Structures*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc. ISBN: 0-13022005-1.
- Witten, I. H. (1977). "An Adaptive Optimal Controller for Discrete-Time Markov Environments". In: *Inform. Control* 34, pp. 286–295.
- Wittgenstein, Ludwig (1953). *Philosophische Untersuchungen*. Frankfurt am Main: Suhrkamp Verlag.
- Woitsch, Robert, Vedran Hrgovic, and Robert Andrei Buchmann (2012). "Knowledge Product Modelling for Industry: The PROMOTE Approach". In: *14th IFAC Symposium on Information Control in Manufacturing 2012*. Information Control Problems in Manufacturing, Volume # 14 | Part# 1, pp. 1208–1213. URL: <http://eprints.cs.univie.ac.at/3913/>.
- Wollnik, M. and E. Grotzla (1986). *Implementierung computergestützter Informationssysteme: Perspektive und Politik informationstechnologischer Gestaltung*. De Gruyter. ISBN: 9783110858495. URL: <https://books.google.de/books?id=HFDoBQAQBAJ>.
- Wongsuphasawat, Kanit et al. (2018). "Visualizing Dataflow Graphs of Deep Learning Models in Tensor-Flow". In: *IEEE Transactions on Visualization and Computer Graphics* 24, pp. 1–12.
- Workflow-Management-Coalition (1998). *Process Definition Interchange Process Model, WfMC TC-1016-M*. Tech. rep. URL: <http://wfmc.org/standards/docs/if19807m.pdf>.
- Wüsteneck, Klaus Dieter (1974). *Philosophisches Wörterbuch*. Vol. 10. 2. Leipzig: VEB Bibliographisches Institut. Chap. Technique, pp. 1209–1210.
- Xu, Tao and Zhidong Xue (Aug. 2019). *NeuroPep—A Comprehensive Resource of Neuropeptides*. URL: http://isyslab.info/NeuroPep/browse_organism?&name=Homo%20sapiens.

- Yanzer Cabral, Anderson R, Marcelo Blois Ribeiro, and Rodrigo Perozzo Noll (2014). “Knowledge management in agile software projects: A systematic review”. In: *Journal of Information & Knowledge Management* 13.01, p. 1450010.
- Ye, Chengxi et al. (2016). “LightNet: A Versatile, Standalone Matlab-based Environment for Deep Learning”. In: *Proceedings of the 2016 ACM on Multimedia Conference*. MM '16. Amsterdam, The Netherlands: ACM, pp. 1156–1159. ISBN: 978-1-4503-3603-1. DOI: <https://doi.org/10.1145/2964284.2973791>. URL: <http://doi.acm.org/10.1145/2964284.2973791>.
- Yoshimura, Yumiko and Edward M Callaway (Oct. 2005). “Fine-scale specificity of cortical networks depends on inhibitory cell type and connectivity”. In: *Nature Neuroscience* 8, 1552 EP –. URL: <http://dx.doi.org/10.1038/nn1565>.
- Yosinski, Jason et al. (2015). “Understanding Neural Networks Through Deep Visualization”. In: *Deep Learning Workshop, International Conference on Machine Learning (ICML)*.
- Young, R. and Asian Productivity Organization (2010). *Knowledge Management Tools and Techniques Manual*. Asian Productivity Organization. ISBN: 9789283324072. URL: <https://books.google.de/books?id=yKGwygAACAAJ>.
- Youdon, Edward and (joint author.) Constantine Larry L (1979). *Structured design: fundamentals of a discipline of computer program and systems design*. English. [Rev. ed.] Includes bibliographies and index. Englewood Cliffs, N.J.: Prentice Hall. ISBN: 0138544719.
- Yu, Rulei and Lei Shi (2018). “A user-based taxonomy for deep learning visualization”. In: *Visual Informatics*. ISSN: 2468-502X. DOI: <https://doi.org/10.1016/j.visinf.2018.09.001>. URL: <http://www.sciencedirect.com/science/article/pii/S2468502X1830038X>.
- Zachman, J. A. (Mar. 1982). “Business Systems Planning and Business Information Control Study: A Comparison”. In: *IBM Syst. J.* 21.1, pp. 31–53. ISSN: 0018-8670. DOI: <https://doi.org/10.1147/sj.211.0031>. URL: <http://dx.doi.org/10.1147/sj.211.0031>.
- Zachman, J. A. (June 1999). “A Framework for Information Systems Architecture”. In: *IBM Syst. J.* 38.2–3, pp. 454–470. ISSN: 0018-8670. <https://doi.org/10.1147/sj.382.0454>. URL: <http://dx.doi.org/10.1147/sj.382.0454>.
- Zachman, John A. (Sept. 1987). “A Framework for Information Systems Architecture”. In: *IBM Syst. J.* 26.3, pp. 276–292. ISSN: 0018-8670. DOI: <https://doi.org/10.1147/sj.263.0276>. URL: <http://dx.doi.org/10.1147/sj.263.0276>.
- Zadeh, Lotfi A. (1965). “Fuzzy Sets”. In: *Information and Control* 8.3, pp. 338–353. DOI: [https://doi.org/10.1016/S00199958\(65\)90241-X](https://doi.org/10.1016/S00199958(65)90241-X). URL: [https://doi.org/10.1016/S00199958\(65\)90241-X](https://doi.org/10.1016/S00199958(65)90241-X).
- Zambrano, Fabian et al. (2016). “Improving patient access to a public hospital complex using agent simulation”. In: *Proceedings of the 2016 Winter Simulation Conference*. T. M. K. Roeder, P. I. Frazier, R. Szcztman, E. Zhou, T. Huschka, and S. E. Chick.
- Zeiler, Matthew D. and Rob Fergus (2013). “Visualizing and Understanding Convolutional Networks”. In: *CoRR* abs/1311.2901. arXiv: 1311.2901. URL: <http://arxiv.org/abs/1311.2901>.
- Zeiler, Matthew D. and Rob Fergus (2014). “Visualizing and Understanding Convolutional Networks”. In: *Computer Vision—ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, pp. 818–833. ISBN: 978-3-319-105901.
- Zelewski, S. (1995). *Petrinetzbasierte Modellierung komplexer Produktionssysteme: Eine Untersuchung des Beitrags von Petrinetzen zur Prozeßkoordinierung in komplexen Pro-*

- duktionssystemen, insbesondere Flexiblen Fertigungssystem. Arbeitsbericht 6. Leipzig: Inst. für Produktionswirtschaft und Industrielle Informationswirtschaft. URL: <https://books.google.de/books?id=hLG7QwAACAAJ>.
- Zelewski, Stephan (2007). *Kann Wissenschaftstheorie behilflich für die Publikationspraxis sein?—Eine kritische Auseinandersetzung mit den “Guideline” von Hevner et al.* Lehner, Franz and Stephan Zelewski In: Wissenschaftstheoretische Fundierung und wissenschaftliche Orientierung der Wirtschaftsinformatik, p. 71–120 GIT Berlin.
- Zentes, J. (1975). *Die Optimalkomplexion von Entscheidungsmodellen: Ein Beitrag zur betriebswirtschaftlichen Meta-Entscheidungstheorie*. Köln: Heymann.
- Zhang, Chi, Jakob Beetz, and Bauke de Vries (2013). “Towards model view definition on semantic level: A state of the art review”. In: *Proceedings of the 20th International Workshop: Intelligent Computing in Engineering*.
- Zhang, T. et al. (2015). “An automatic and effective parameter optimization method for model tuning”. In: *Geoscientific Model Development* 8.11, pp. 3579–3591. DOI: <https://doi.org/10.5194/gmd-8-3579-2015>. URL: <https://www.geosci-model-dev.net/8/3579/2015/>.
- Zhang, Zuopeng and Shankar Sundaresan (2010). “Knowledge markets in firms: knowledge sharing with trust and signalling”. In: *Knowledge Management Research & Practice* 8.4, pp. 322–339.
- Zhou, Bolei et al. (2016). “Learning deep features for discriminative localization”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921–2929.
- Zimmer, G. (1987). *Selbstorganisation des Lernens: Kritik der modernen Arbeitserziehung*. 11. Lang. URL: <https://books.google.de/books?id=RaYwAAAACAAJ>.
- Zintgraf, Luisa M et al. (2017). “Visualizing deep neural network decisions: Prediction difference analysis”. In: *arXiv preprint arXiv: 1702.04595*.
- Zschocke, D. (1995). *Modellbildung in der Ökonomie: Modell—Information—Sprache*. Vahlen. Handbücher der Wirtschafts- und Sozialwissenschaften. Vahlen. ISBN: 9783800619627. URL: <https://books.google.de/books?id=-yKePAAACAAJ>.
- Zubin, Joseph (1938). “A technique for measuring like-mindedness.” In: *The Journal of Abnormal and Social Psychology* 33.4, p. 508.
- Zwicky, F (1966). *Entdecken, Erfinden, Forschen im morphologischen Weltbild*. Munich: Droemer Knaur.