

Architecting Model Driven System Integration in Production Engineering

Yujiang Li

Doctoral Thesis
KTH Royal Institute of Technology
Engineering Sciences
Department of Production Engineering
Stockholm, Sweden, 2017

ISBN 978-91-7729-416-0

TRITA-IIP-17-07

ISSN 1650-1888

Abstract

System integration is a key enabler to maximize information value in an engineering context. The valuable information is normally represented by information models which play a decisive role in the implementation of system integration. The information models are designed to efficiently and effectively capture, process and communicate information among different functional and business units. However, use of the information models in implementing system integration is challenged by insufficient support from current settings of modeling architectures. This situation calls for new strategies to ease the use of information models.

To address this challenge, this study presents a new twofold solution: Model driven system integration. It includes 1) a modeling architecture to guide the development of information models and 2) an integrated implementation process to guide the use of information models. Thus, this work improves practical applicability of an information model in its entire modeling lifecycle.

The results contribute not only to the performance of modeling practices but also to improved understanding of information modeling in system integration. Implementation contexts and implementation models are introduced to develop an implementation-oriented modeling architecture. Further, the potential of information models as a knowledge base to support implementation practices is identified.

To concretely discuss behaviors and structures of information models, this study adopts ISO 10303 and the related standards as major references of existing information models.

Case studies on model driven system integration validate this research in scenarios concerning kinematic modeling, kinematic error modeling, cutting tools classification and product catalogue modeling. Model driven system integration exhibits high efficiency in implementation, enhanced interoperability and increased value of information models.

Keywords

System architecture, system integration, information model, ISO 10303, application context, implementation context, implementation model.

Sammanfattning

Systemintegration är väsentligt när man vill maximera informationsvärdet i ingenjörsmässiga sammanhang. Värdefull information representeras normalt av informationsmodeller som spelar en avgörande roll i genomförandet av systemintegration. Dessa informationsmodeller är utformade för att effektivt fånga, bearbeta och förmedla information mellan olika funktionella enheter och affärsheter. Emellertid stödjs inte användningen av informationsmodeller tillräckligt vid genomförandet av systemintegration i nuvarande modelleringssarkitekturen. Detta gör att integration försvaras eller inte blir av, vilket kräver nya strategier för att underlätta användningen av informationsmodeller.

För att ta itu med denna utmaning presenterar studien en ny lösning: Modelldriven systemintegration. Lösningen innehåller två delar: 1) en modelleringssarkitektur som vägleder hur man kan utveckla en informationsmodell och 2) en integrerad process för att vägleda hur man kan använda informationsmodeller. På detta sätt bidrar arbetet till en pragmatisk tillämpning av informationsmodeller i hela modelleringens livscykel. Studien bidrar inte bara till modelleringssmetoden som ett resultat, utan ger också ökad förståelse för informationsmodellers roll och användning inom systemintegration. Kontextbeskrivningar och modeller för genomförande introduceras för att utveckla modelleringssarkitekturen. Vidare så har potentialen av att använda informationsmodeller som kunskapsbaser identifierats som ett sätt att stödja genomförandet.

För att konkret diskutera beteenden och strukturer av informationsmodeller, använder denna studie ISO 10303 och tillhörande standarder som viktiga referenser av befintliga modelleringssarkitekturen. Fallstudier inom modelldriven systemintegration används för validering, i scenarier för kinematisk modellering, modellering av kinematiska fel, klassificering av skärande verktyg och produktkatalogmodellering. Modelldriven systemintegration uppvisar hög effektivitet vid genomförandet, förbättrad interoperabilitet och ökat värde av informationsmodeller.

Acknowledgement

At the end of this graduate study, I am extremely proud to work with many colleagues and friends during my journey here. The dissertation would not happen without the unbelievable patience and wonderful support of my major supervisors, Prof. Torsten Kjellberg and Dr. Mikael Hedlind. They helped me great to better organize and formulate my ideas. Prof. Lars Mattsson and Dr. Gunilla Sivard are also great supervisors who have, throughout my study, mentored and supported me.

I am honored to work with many talented people offering enlightened advices and content to my study, including Dr. Danfang Chen (陈丹芳), Magnus Lundgren, Prof. Lihui Wang (王力翠), Dr. Rong Su (苏榕), Qiuling Huang (黄秋灵), Dr. Thomas Lundholm, Christer Wickman and Andreas Bergstrand. They added much to the depth and broadness of my research.

I would like to thank the cool Chinese fellows for their amazing support to my work and my living in Sweden, to whom I am greatly indebted: Dr. Wei Ji (计伟), Dr. Xu Liu (刘旭), Dr. Wei Wang (王伟), Dr. Dawei Liu (刘大炜), Dr. Qilin Fu (付麒麟), Hongyi Liu (刘弘逸), Bitao Yao (姚碧涛), Dr. Yue Meng (孟悦) and Dr. Vicent Wang (王曦).

I am grateful to many colleague “IIPers” for their support, including Assoc. Prof. Daniel T. Semere, Assoc. Prof. Lasse Wingård, Per Johansson, Navid Shariat Zadeh, Lars Lindberg, Dr. Antonio Maffei, Prof. Mauro Onori, Dr. João Ferreira, Johan Pettersson, Anna Wiippola, Sarah Golibari and Anna Eklund.

I would like to acknowledge Prof. Zhizhong Li (李志忠), Prof. Pei-Luen Patrick Rau (饶培伦) and Prof. Kaibo Wang (王凯波), from Tsinghua University, Beijing, who inspired me to be devoted to academic research.

I have been fortunate to have my beloved parents, who have been always there with extraordinary kindness and goodwill and my dear friends in China and America, Dr. Yan Deng (邓彦), Xiaolu Dong (董晓璐), Dr. Ji-yuan Gao (高基元), Prof. Ruiwei Jiang (江瑞威), Jianfeng Liao (廖剑锋), Xiaoting Lin (林晓婷), Fei Song (宋霏), Guanxiang Wan (万观祥), Rongwen Xie (谢榕文) and Prof. Ming Zeng (曾鸣).

Abbreviations

AAM	Application Activity Model
AEC	Architecture, Engineering and Construction
AIM	Application Integrated Model
AM	Application Module
AM MVC	Application Model MVC
AP	Application Protocol
API	Application Programming Interface
ARM	Application Reference Model
ASD	Automotive, aerospace and defense industry
ASME	American Society of Mechanical Engineers
BIM	Building Information Model
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CAx	Computer-Aided technology
CNC	Computer Numerical Control
DFBB	Digital Factory Building Blocks
GD&T	Geometric Dimensioning and Tolerancing
GPS	Geometrical Product Specification
GUI	Graphical User Interface
FBOP	Feature Based Operation Planning
HCD	Human-Centered Design
HCI	Human-Computer Interaction
IA	Information Architecture
IaaS	Infrastructure as a Service
IDEFo	Integrated computer aided manufacturing DEFinition for Function modeling
IDM	Information Delivery Manual
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IFC	Industry Foundation Classes
Impl.	Implementation
ISO	International Organization for Standardization
IT	Information Technology
IxD	Interaction Design
JAR	Java ARchive
LCIM	Level of Conceptual Interoperability Model
Mfg.	Manufacturing
MIM	Module Interpreted Model

MPQP	Model-driven Process and Quality Planning
MVC	Model-View-Controller
MVD	Model View Definition
MVP	Model-View-Presenter
OOSE	Objective-Oriented Software Engineering
PaaS	Platform as a Service
PFMEA	Process Failure Mode and Effects Analysis
PLCS	Product Life Cycle Support
PLib	Parts Library (ISO 13584)
PLM	Product Lifecycle Management
PPR	Product-Process-Resource
RPN	Risk Priority Number
SaaS	Software as a Service
SDAI	Standard Data Access Interface
SOA	Service Oriented Architecture
SoS	System of Systems
STEP	STandard for the Exchange of Product data (ISO 10303)
STEP-NC	NC programming language extending STEP (ISO 14649)
TTM	Time To Market
UI	User Interface
UML	Unified Modeling Language
UoF	Unit of Functionality
UX	User Experiences
VBA	Visual Basic for Applications
XaaS	Everything as a Service
XML	eXtensible Markup Language
XP	eXtreme Programming

List of appended publications

Paper A

Li, Y., Hedlind, M. and Kjellberg, T. (2012a). Kinematic error modeling based on STEP AP242, *Proceedings of the 1st CIRP Sponsored Conference on Virtual Machining Process Technology*, Montreal.

Paper B

Li, Y., Hedlind, M., Kjellberg, T. and Sivard, G. (2013a). Cutting tool data representation and implementation based on STEP AP242, *Smart Production Engineering*, pp. 483-492, Berlin/Heidelberg: Springer-Verlag. DOI: [10.1007/978-3-642-30817-8_47](https://doi.org/10.1007/978-3-642-30817-8_47).

Paper C

Li, Y., Huang, Q., Hedlind, M., Sivard, G., Lundgren, M. and Kjellberg, T. (2014a). Representation and exchange of digital catalogues of cutting tools. *Proceedings of ASME 2014 International Manufacturing Science and Engineering Conference*, Detroit. DOI: [10.1115/MSEC2014-4131](https://doi.org/10.1115/MSEC2014-4131).

Paper D

Li, Y., Chen, D., Kjellberg, T. J. and Sivard, G. (2014b). User friendly development architecture for standardised modelling: STEP toolbox, *International Journal of Manufacturing Research*, Vol. 9, No. 4, pp. 429-447. DOI: [10.1504/IJMR.2014.066663](https://doi.org/10.1504/IJMR.2014.066663).

Paper E

Li, Y., Hedlind, M., Kjellberg, T. and Sivard, G. (2014c). System integration for kinematic data exchange, *International Journal of Computer Integrated Manufacturing*, Vol. 28, No. 1, pp. 87-97. DOI: [10.1080/0951192X.2014.941937](https://doi.org/10.1080/0951192X.2014.941937).

Paper F

Li, Y., Hedlind, M., & Kjellberg, T. (2015b). Usability Evaluation of CAD-CAM: State of the Art, *Procedia CIRP*, Vol. 36, pp. 205-210. DOI: [10.1016/j.procir.2015.01.053](https://doi.org/10.1016/j.procir.2015.01.053).

Authored publications not appended

Li, Y., Hedlind, M. and Kjellberg, T. (2011). Implementation of kinematic mechanism data exchange based on STEP, *Proceedings of the 7th CIRP-Sponsored International Conference on Digital Enterprise Technology*, pp. 152-159, Athens. ISBN: 978-960-88104-2-6.

Hedlind, M., Klein, L., Li, Y. and Kjellberg, T. (2011). Kinematic structure representation of products and manufacturing resources, *Proceedings of the 7th CIRP-Sponsored International Conference on Digital Enterprise Technology*, pp. 340-347, Athens. ISBN: 978-960-88104-2-6.

Li, Y., Su, R., Hedlind, M., Ekberg, P., Kjellberg, T. and Mattsson, L. (2012b). Model based in-process monitoring with optical coherence tomography, *Procedia CIRP*, Vol. 2, pp. 70-73. DOI: [10.1016/j.procir.2012.05.042](https://doi.org/10.1016/j.procir.2012.05.042).

Li, Y. (2013b). Development architecture for industrial data management, *Licentiate Thesis*, Stockholm: Royal Institute of Technology. ISBN: 978-92-7502-903-1.

Contents

Chapter 1	Introduction	1
1.1	Research scope	2
1.1.1	Model driven system integration	2
1.1.2	Architecting	5
1.2	A new modeling architecture	7
1.2.1	Pragmatic concern for implementers	7
1.2.2	Development of a new modeling architecture.....	9
1.3	An integrated implementation process.....	13
1.3.1	Two types of unused potential of information models.....	13
1.3.2	Make use of the potential in process integration.....	15
1.4	Main contributions	17
1.4.1	Identification of the root cause of challenges.....	17
1.4.2	Solutions.....	19
1.4.3	New concepts.....	20
Chapter 2	Frame of reference.....	23
2.1	Interoperability.....	23
2.1.1	Interoperation as a challenge.....	24
2.1.2	How to understand interoperability	26
2.1.3	Semiotic aspects of interoperability	27
2.1.4	Pragmatics: A focus of this study	33
2.2	Standardized information models	36
2.2.1	Application contexts and information requirements	37
2.2.2	The STEP modeling architecture	43
2.2.3	Extensibility and portability of STEP	49
2.2.4	High extensibility is preferable	52
2.3	A model driven approach	54
2.3.1	Contextual proprietary information models	54
2.3.2	Composable standardized generic information models... ..	56
2.3.3	Combination of contextual and composable solutions....	57
2.4	System architectures	58

2.5 Information architectures	62
Chapter 3 A new modeling architecture	65
3.1 Envisioned pragmatic interoperability	65
3.2 Application contexts: A pitfall	68
3.2.1 Application contexts ignoring implementers	68
3.2.2 A different implementation workflow.....	69
3.2.3 Increasing complexity of modeling architectures.....	71
3.3 Implementation contexts: The new concern.....	72
3.3.1 Defining implementation contexts.....	72
3.3.2 Use of implementation contexts	78
3.3.3 Discussions of implementation contexts	81
3.4 Implementation models.....	82
3.4.1 Semiotics of information models.....	82
3.4.2 Encapsulation	89
3.4.3 Information requirements.....	90
3.4.4 Creation of an implementation model.....	94
3.4.5 An exemplification.....	96
3.5 Layered architectures	99
3.5.1 The composable layer	101
3.5.2 The contextual layer	102
3.5.3 Closed layered and open layered architectures	103
3.6 Discussions.....	105
Chapter 4 A model driven implementation process	109
4.1 How to use information models	110
4.1.1 Current limited use of information models	110
4.1.2 Envisioned use of information models.....	112
4.1.3 Development of an integrated process for the envisioned use of information models	113
4.2 The essence of software implementation activities.....	117
4.2.1 Stages of software implementation	118
4.2.2 Common software implementation approaches	121
4.3 Model based software implementation.....	122
4.3.1 Involvement of the HCD paradigm	123
4.3.2 In-depth participation of domain experts	127

4.4 Process integration.....	128
4.5 The scope stage	134
4.5.1 User studies	135
4.5.2 Visions.....	137
4.6 The conception stage	140
4.6.1 Scenarios	142
4.6.2 Requirements.....	144
4.7 The framework stage	149
4.7.1 Use cases.....	150
4.7.2 A design framework	152
4.8 The abstraction stage	158
4.8.1 Process models.....	160
4.8.2 Data models	163
4.8.3 Wireframe design	164
4.9 The construction stage.....	167
4.9.1 Program design.....	167
4.9.2 Programming and testing	171
Chapter 5 Case studies	173
5.1 Realization of an implementation model: STEP Toolbox..	173
5.2 Kinematic data management.....	177
5.3 Cutting tool data management.....	180
Chapter 6 Conclusions.....	183
6.1 Discussion	184
6.2 Limitations	187
6.3 Future work.....	189
Definitions of relevant terms.....	195
Basic terms	195
Terms about information models.....	196
Qualities pursued in this study	197
Bibliography	199
Appended papers.....	213

Figures and tables

Figure 1.1 A standardized contractor in system integration.	3
Figure 1.2 Model driven system integration: An interdisciplinary topic....	4
Figure 1.3 Architectures and stakeholders in a context of the model driven system integration.	6
Figure 1.4 Information models to facilitate semantic interoperability. Both data schemas and data sets are information models.	7
Figure 1.5 Implementers interpret a part of modeling architecture to develop applications conforming to the data schemas.	8
Figure 1.6 An envisioned situation of model development and software development.....	11
Figure 1.7 Publications relative to the layered architecture.....	12
Figure 1.8 Results of the complex modeling architectures	14
Figure 1.9 Software implementation utilizes different types of information models in a modeling architecture.	15
Figure 1.10 An integrated process for model driven system integration..	15
Figure 1.11 The conventional modeling architecture of ISO 10303 STEP, based on textual description by Anderl and Wasmer (1997).....	18
Figure 1.12 Thesis structure.....	20
Figure 1.13 An application as an intermediate between stakeholders and information models.	21
Figure 2.1 Reuse of existing data by implementers and users.....	25
Figure 2.2 A situation where programs (i.e. application software) control users, presented by Richard Stallman at TED (2014).....	26
Figure 2.3 “A sign-situation, or a process of semiosis” (drawing based on Morris, 1939).....	28
Figure 2.4 Four cornerstones and three connections of the framework proposed by Lindland, et al. (1994).....	29
Figure 2.5 Model based computerized information systems (reproduction based on Hofmann, 2003).....	31
Figure 2.6 Three levels of interoperability concerning this study.	34
Figure 2.7 Computer interpretation of information models (data sets) needs human interpretation of information models (data schemas).	35
Figure 2.8 Application context used in software implementation.	38

Figure 2.9 The application contexts defining information models	38
Figure 2.10 A traditional view of information requirements as a subset of requirements to be implemented in information models.....	40
Figure 2.11 A data schema and a data set.....	40
Figure 2.12 Application context, the current basis for application development.	41
Figure 2.13 Application context, the current basis for both application development and information model development.....	42
Figure 2.14 Application context used for information model development.	43
Figure 2.15 The traditional modeling architecture of ISO 10303 STEP... 44	
Figure 2.16 Artifacts (rectangles) in STEP modeling architecture useful for implementers.	45
Figure 2.17 Software development based on the STEP modeling architecture.....	46
Figure 2.18 A workflow to implement information model standards based on STEP modeling architecture.....	48
Figure 2.19 Information models from different domains are used by applications focusing on applying multiple views of product data.. 49	
Figure 2.20 A data schema and a data set.....	50
Figure 2.21 Extending the data schema based on information requirements from new application context.	51
Figure 2.22 Porting the information model to new application.	51
Figure 2.23 Reusable building blocks to compose an information model.	53
Figure 2.24 Contextual abstraction and composable abstraction.	56
Figure 3.1 Pragmatic interoperability that should be addressed by information models.	66
Figure 3.2 Pragmatic interoperability of an information model facilitates portability, usability and implementability with implementers as direct users.....	68
Figure 3.3 The current situation: The application context forms a basis for application implementation as well as model development.....	69
Figure 3.4 The traditional way to implement an application conforming to a STEP data schema.....	70

Figure 3.5 Evolution of the modeling architectures, based on ISO (2016c).	71
Figure 3.6 The usage of the implementation context.	72
Figure 3.7 Implementation context, a shifted basis where information model development begins.	73
Figure 3.8 User groups and their positions in an implementation context.	76
Figure 3.9 Tasks performed with CADCAM systems, based on (Makris, et al. 2014; Zeid, 1991).	77
Figure 3.10 In the information model development process.	79
Figure 3.11 UI of Siemens NX 7.5 visualizes a kinematic link in relation to its corresponding geometry.	84
Figure 3.12 UI of STEP-NC Machine 9.38 visualizes motion simulation.	85
Figure 3.13 UI of a STEP AP242 model translator developed in this study.	85
Figure 3.14 Components in an implementation context.	91
Figure 3.15 Information requirements based on implementation contexts.	93
Figure 3.16 An example for instantiating a kinematic link, excerpted from an AP242 data set.	97
Figure 3.17 Diagrammatic information requirements for an implementation model for kinematic data exchange.	98
Figure 3.18 A class diagram for kinematics in the STEP Toolbox API for Java.	99
Figure 3.19 The layered modeling architecture.	100
Figure 3.20 A closed layered architecture.	104
Figure 3.21 An open layered architecture.	104
Figure 3.22 Decoupling to reduce dependency.	107
Figure 4.1 A traditional mapping process for using a data schemas.	110
Figure 4.2 Current integration of the information model mapping with the implementation process.	111
Figure 4.3 Closely integrating the mapping process with the implementation process.	113
Figure 4.4 A general mapping pattern of engineering activities.	114
Figure 4.5 Models are used to translate reality to products.	114

Figure 4.6 Software implementation supports (physical) product realization in manufacturing industry: The scope of this chapter..	115
Figure 4.7 Apply data schemas for system integration.....	116
Figure 4.8 Information models are used in implementation.	116
Figure 4.9 Information models have potential to support implementation.	117
Figure 4.10 Categorization of engineering design activities (McNeill, et al., 1998).	118
Figure 4.11 Segments of design protocols (Akin and Lin, 1995).....	118
Figure 4.12 The waterfall software implementation style aimed at eliminating unwanted iterations (Royce, 1970).	119
Figure 4.13 The incremental development style and the prototyping development style.	120
Figure 4.14 Microsoft “Synch and stabilize” process (Cusumano and Selby, 1997).	122
Figure 4.15 The 4+1 view model of software architecture.	123
Figure 4.16 Four activities of interaction design (Sharps, et al., 2007).	124
Figure 4.17 The star model (Hix and Hartson, 1993).	125
Figure 4.18 The standardized HCD process (ISO, 2010b).....	125
Figure 4.19 Usability engineering lifecycle (Mayhew, 1999).	126
Figure 4.20 A generalized process of software implementation.	127
Figure 4.21 Processes need to be integrated.	128
Figure 4.22 The information model mapping process underpin a process to implement software at different levels of abstraction.	129
Figure 4.23 Integration of the processes.....	130
Figure 4.24 Stages and activities of the implementation process.	131
Figure 4.25 Activities of <i>Model driven software implementation process for system integration</i> and the parallel processes.....	133
Figure 4.26 The highest level of the activity models defined in ISO 14649.	134
Figure 4.27 An example of proto-persona for a user of risk assessment applications.....	135
Figure 4.28 The user model for users of kinematic data exchange.	137
Figure 4.29 A vision proposal for a risk assessment system.	139
Figure 4.30 A vision proposal for a digital catalogue exchange system.	140

Figure 4.31 A vision proposal for kinematic data exchange. (“KIBOS” denotes KTH Implementation Based On STEP)	140
Figure 4.32 A high-level reference model specified in ISO 14649 (author reproduction based on ISO, 2003b).....	141
Figure 4.33 Requirements of risk assessment application.	147
Figure 4.34 Requirements of kinematic data exchange.....	148
Figure 4.35 An activity model related with machining operations (author reproduction based on ISO 14649 STEP-NC).	149
Figure 4.36 A reference model related with machining operations (author reproduction based on ISO 14649 STEP-NC).	150
Figure 4.37 a high-level conceptual model for data elements.	157
Figure 4.38 An excerpt of a data schema for STEP p105 ed2 (author reproduction based on ISO, 2014a).....	159
Figure 4.39 A passive MVP pattern.....	160
Figure 4.40 The classic concept of MVC.	161
Figure 4.41 Views and controllers are decoupled (Fowler, 2006).	162
Figure 4.42 AM MVC (Sharan, 2015).	162
Figure 4.43 MVP (Potel, 1996).	162
Figure 4.44 Passive MVP (Sharan, 2015)	162
Figure 4.45 A flow chart to describe the process model.	163
Figure 4.46 A class diagram to model the domain concepts.	164
Figure 4.47 A wireframe design for identifying a specific risk item.	166
Figure 4.48 A wireframe design for specifying risk items.	166
Figure 4.49 An interaction diagram for an implementation integrating with existing CAx applications.	168
Figure 4.50 An interaction diagram for an implementation as standalone application integrated with users’ existing workflow.	169
Figure 4.51 A general interaction diagram based on MVC for both types of implementations.....	170
Figure 5.1 A unique process to implement STEP-based standards.	174
Figure 5.2 The modeling architecture using STEP Toolbox API.	175
Figure 5.3 Documented STEP Toolbox API by Javadoc.	176
Figure 5.4 A flow chart for an integrated application to export a data set with full integration of kinematic data and geometric data.....	178
Figure 5.5 A process design enhanced with STEP Toolbox API.	178
Figure 5.6 Process design of a kinematic data translator for CATIA.....	179

Figure 5.7 Mapping of hierarchy	181
Figure 5.8 Prototypes for catalogue data exchange.	181
Figure 6.1 Implementers develop applications based on data schemas to make data sets accessible for application users.	185
Figure 6.2 “Paradoxical loop” of implementing information models (author reproduction based on Kiviniemi, et al., 2008).....	190
Figure 6.3 Discipline models and aggregate models in different stage of a product lifecycle (Van Berlo, et al., 2012).	193
Table 2.1 Different requirements on information models by extensibility and portability.	54
Table 3.1 Specifications to facilitate semiotic aspects of information models standardized in the framework of STEP.....	83
Table 3.2 An excerpt of a p21 (conforming to ISO 10303-21) data set....	84
Table 3.3 An example of an ISO 10303 data schema (a part of the long- form schema of STEP AP242).	88
Table 3.4 Principles usable for representing information requirements. 92	
Table 3.5 Specifications facilitating semiotics of implementation models.	94
Table 4.1 Working reengineering on a use case to specify risk assessment.	151
Table 4.2 Design framework of risk assessment.....	155
Table 4.3 Design framework of kinematic data exchange.	156

Chapter 1

Introduction

*The greatest challenge to any thinker is stating the problem
in a way that will allow a solution.*

- Bertrand Russell

Information technology (IT) has been serving engineers for decades in the field of product realization. A central artifact for product realization is product data¹ stored, communicated, and processed in the form of models. In this respect, human-model interaction becomes a dominant activity of the product realization process. It is in step with the fashion of human beings to live in the Information Age: “Man the food-gatherer reappears incongruously as information-gatherer. In this role, electronic man is no less a nomad than his paleolithic ancestors” (MeLuhan, 1964).

When engineers are benefited from computerized information systems², effective use of the product data becomes problematic. It is usual that the computerized information systems, instead of engineers, take control of how the product data can be used. A tool-driven manner³ (more details in Section 2.1.1) illustrates this phenomenon where the computerized information systems limit how users can interact upon product data.

To reuse and integrate product data, the core activity is system integration (Section 1.1.1) which leverages standardized information models⁴ (Section 2.2) as the major facilitator. The information models governed by

¹ Product data: “A representation of information about a product in a formal manner suitable for communication, interpretation, or processing by human beings or by computers” (ISO, 1994), including data for lifecycle management, manufacturing, design, etc.

² Computerized information system: A “human-machine system that provides information to support the operational, managerial, analytic, and decision-making functions” based on the use of computers (Krogstie, 2012; Falkenberg, et al., 1998).

³ Tool-driven manner: A problematic way to complete engineering goals greatly depending on tools in use with limited functions and content (Section 2.1.1).

⁴ Information model: “A formal model of a bounded set of facts, concepts or instructions to meet a specified requirement” (ISO, 1994).

modeling architectures could greatly impact implementation performance of system integration. Whoever using the information models will face the nature of its architecture—sophisticated semantic representations in an interdisciplinary context. It brings both troubles and opportunities. This thesis analyzes deep-down logic behind troubles of implementers and puts forth strategies for efficient system integration taking advantages of information models.

1.1 Research scope

As an interdisciplinary task, system integration based on information models is a difficult but crucial challenge for both software engineers and production engineers. This section explains the research scope with main concepts where the challenge need to be resolved. Two major concepts form the research scope, which are also indicated in the thesis title, i.e. model driven system integration and architecting activities.

1.1.1 Model driven system integration

System integration is the activity to meet the challenge of interoperability¹ (Section 2.1), as a process to progressively assemble “system components into the whole system” (ISO/IEC/IEEE, 2010). The components may be existing systems, envisioned systems or legacy systems. Data output by diverse systems may be fundamentally distinct and create obstacles between the systems. System integration is intended to fix these obstacles to satisfy requirements that cannot be met by these individual systems. It is the key path toward enterprise integration enabling connection between functional entities (i.e. devices, applications and people) beyond organizational boundaries and technical boundaries (Sherif, 2009).

To implement system integration, a type of standardized contractors (also known as protocols) should be identified (Figure 1.1). For CAx (Computer-Aided technology), product data is usually formalized as information models. Particularly, standardized generic information models (Section 2.2) are useful as the contractors to communicate product data. The major

¹ Interoperability: “The ability of two or more systems or components to exchange information and to use the information that has been exchanged” (ISO/IEC/IEEE, 2010).

reference of information models chosen in this thesis is the widely accepted standard ISO 10303 STEP (STandard for the Exchange of Product data).



Figure 1.1 A standardized contractor in system integration.

Although the illustration looks very simple, a standardized generic information model is not the “one-stop” solution for system integration. Making an information model usable for all the systems could be a daunting task. Generally speaking, software implementation has “no silver bullet” (Brooks, 1987) because software engineers often dealt with requirements from other disciplines. The interdisciplinary nature is magnified when leveraging standardized information models.

To relatively ease the task, formal information models provide established modeling architectures: Schemas and methodologies (for representation, interpretation and processing) are specified as a package (more details in Section 2.2). Nevertheless, it is still not easy for implementers to learn and use such kinds of modeling architectures with domain-specific sign systems.

To support implementers’ work for system integration based on information models, this study proposes a solution named model driven system integration. It is based on the model driven approach (more details in Section 2.3), i.e. an approach using information models to integrate product data from different sources, to create a productive environment and to enhance business competitiveness. It implies that the information models act as a driver to exploit the potential of information technology for product realization.

As illustrated (Figure 1.2), the practices of model driven system integration involve at least three subject areas. Between the subject areas, architectures (Section 2.4) help to make use of knowledge from different subject areas and to manage complexity due to the interdisciplinary nature. The involved subject areas and architectures can be defined as following:

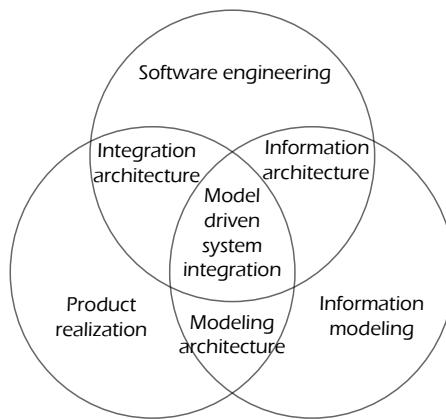


Figure 1.2 Model driven system integration: An interdisciplinary topic.

- Software engineering: “An engineering discipline that is concerned with all aspects of software production”. (Sommerville, 2011)
- Information modeling: Theories, approaches, tools and processes of development and use of information models (Section 2.2)¹.
- Product realization: An engineering discipline studying all activities and functions (planning, controlling, operation and organization) directly contributing to making of goods².
- Integration architecture: Structure, relations and rationales of system components for system integration (Section 2.1), decided at an architectural level.
- Information architecture: “(Human-centred) structure of an information space and the semantics for accessing required task objects, system objects and other information” (ISO/IEC, 2010, Section 2.5).

¹ This definition is based on the definition of “modelling” by Knuuttila (2011), i.e. “construction and use of models”. Note that von Euler-Chelpin (2008) has also defined information modeling with the focus only on construction: “The activity of identifying, relating, and structuring the information types that need to be managed into an information model.”

² This definition reuses the definition of “manufacturing”, in a broad sense, by CIRP (2004): “The entirety of interrelated economic, technological and organizational measures directly connected with the processing/machining of materials, i.e. all functions and activities directly contributing to the making of goods.”

- Modeling architecture: An architecture to specify development and use of an information model.
- Model driven system integration: An integration practice to facilitate interoperability between disparate systems based on the model driven approach (Section 2.3).

As illustrated, the model driven system integration builds a connection between modeling architectures and software engineering. This study is to change the way of development (Chapter 3) and use (Chapter 4) of information models, i.e. the modeling architectures, for the purpose of improve implementation performance of system integration.

1.1.2 Architecting

Architectures typically aim at managing complexity of interdisciplinary engineering tasks and communicating design ideas during system development. The architectures in this thesis indicate the interdisciplinary nature of implementing model driven system integration and the concern for human beings who are stakeholders of the implementation.

As suggested in Figure 1.2, architecting is the activity to resolve the interdisciplinary complexity with knowledge of many domains involved. An *architecture* (Section 2.4) is an engineering artifact to manage the application of knowledge to build a system (ISO/IEC/IEEE, 2011b). The three overlaps between each pair of circles in Figure 1.2 indicate architectures due to cross-domain integration, which are relevant for model driven system integration, i.e.:

- The integration architecture explores the best way to integrate computerized information systems for product realization.
- The information architecture utilizes and implements information models in software systems.
- The modeling architecture supports use of information models for product realization.

The three architectures are created to resolve complexity from two corresponding subject areas, and also strongly influenced by the third subject area. That is, the integration architecture heavily relies on utilization of information models; the information architecture should reflect requirements from production engineering; the modeling architecture is a facilitator for software implementation.

This thesis uses architecting also to express concern for human beings. Any system has many concerned stakeholders, e.g. end-users, developers, maintainers, customers. Development and use of a system implies application of domain knowledge of different stakeholders in different perspectives. Few people can understand all the relevant domains. Therefore, the architectures should address concerns of different stakeholders and integrate knowledge from different domains.

The architectures mentioned above are practically intermediate artifacts to support the work of different stakeholders whose final goals are applications to satisfy application users. How the stakeholders including application users work with the architectures and the final applications is illustrated in Figure 1.3. Indeed, application users are central stakeholders and applications are central artifacts to be produced. The user needs should be satisfied by the produced applications. However, all the architectures (in the middle of Figure 1.3) should serve the model developers and implementers (including application designers and constructors).

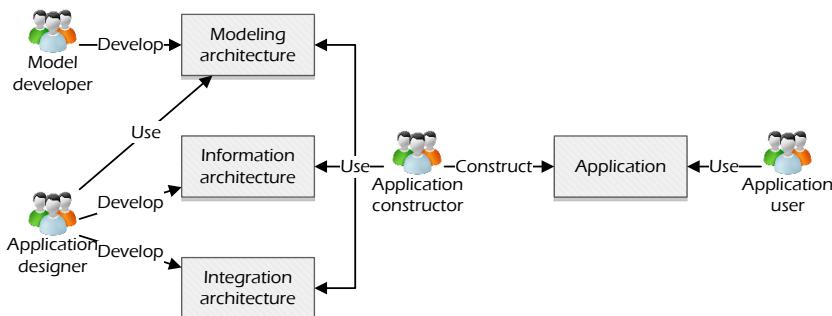


Figure 1.3 Architectures and stakeholders in a context of the model driven system integration.

Among the stakeholders, this study pays special attention to the implementers (application designers and application constructors) since they are direct users of information models. The mission of this study is to clarify their needs in relation to the modeling architectures and to reflect their needs in model driven system integration to enhance implementation performance: Firstly, to explore what is the best modeling architecture for the

implementers (Chapter 3); secondly, to clarify how implementers create integration architectures and information architectures utilizing a modeling architecture (Chapter 4).

1.2 A new modeling architecture

This section is a brief introduction of Chapter 3. The first part (Section 1.2.1) introduces necessary background knowledge, a challenge of implementation efficiency and a current incomplete focus of application contexts¹ as the root cause of the challenge. The second part (Section 1.2.2) introduces how to resolve the challenge by dealing with the root cause, i.e. taking implementers into account when developing modeling architectures.

1.2.1 Pragmatic concern for implementers

For applying information models in system integration, implementers suffer from evident inefficiency (Section 3.2 and Paper D), which hinders wide practical use of many good information standards. Before proposing any solution to this problem, this study begins with locating the root cause, just as Joe Biden stated: “If you don’t understand what the cause is, it’s virtually impossible to come up with a solution”.

Currently, standardized generic information models can achieve semantic interoperability² very well for computer interpretation, e.g. with the widely accepted ISO 10303. The achievement is based on computer interpretable data sets³ governed by semantically comprehensive data schemas⁴ (Figure 1.4).

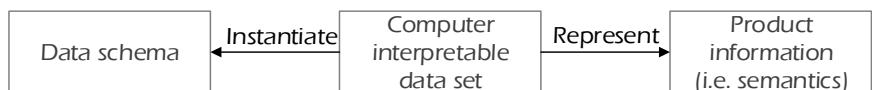


Figure 1.4 Information models to facilitate semantic interoperability.
Both data schemas and data sets are information models.

¹ Application context: An environment where product data is used in a specific application (more details in Section 2.2.1).

² Semantic interoperability: The ease to interpret meanings of exchanged messages by systems in an interoperation scenario (more details in Section 2.1.3).

³ Data set: An information model instantiating what defined in a data schema (more details in Section 2.2).

⁴ Data schema: An information model to represent types of information in a particular context (more details in Section 2.2).

However, successful computer interpretation of an information model cannot guarantee successful use of it (more details in Section 2.1.4). Human interpretation is inevitably involved because implementers should use data schemas and related data interfaces. Thus, the modeling architectures are expected to support implementers as well (Figure 1.5). This support is currently insufficient in popular standardized information models (e.g. ISO 10303), leading to inefficiency of implementation. This inefficiency could affect quality of an information model because use of any information model begins with implementation.

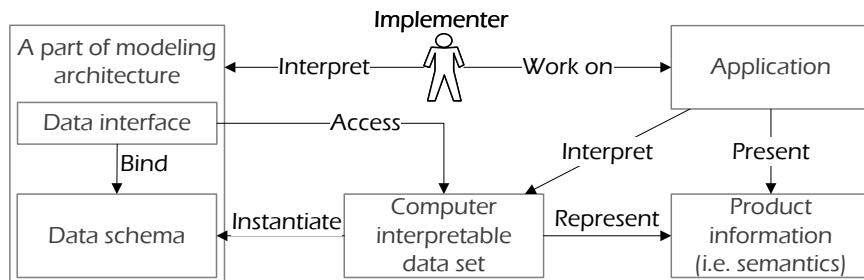


Figure 1.5 Implementers interpret a part of modeling architecture to develop applications conforming to the data schemas.

The insufficient support could be attributed to application contexts. Typically, the application contexts are used by implementers to design and develop application software. The application contexts are also used by model developers as an initial step (more details in Section 2.2.1). Nevertheless, when model developers only see the application contexts, primary users (i.e. implementers) of model developers' work (i.e. data schemas) are neglected (more details in Section 3.2). For instance, implementation activities are not included in STEP Application Activity Models (AAMs) which represent the application contexts (more details in Section 2.2.2).

For standardization, translation from the application contexts to the data schemas should be formalized (more details in Section 2.2.1 and 2.2.2), which has negative effects: The focus drives complicated mapping mechanisms and complicated hierarchical modeling structures (more details in

Section 3.2). These complications may be understandable for model developers, but is uneasy to handle for normal implementers.

1.2.2 Development of a new modeling architecture

Hence, changes should be made for initializing modeling architectures based on application contexts. This study proposes implementation contexts¹ (Section 3.3) as a new basis for modeling architectures. Section 3.3 describes what this new concept is and how it is used in the development of information models. The implementation context implies a necessary goal for a modeling architecture, i.e. to boost implementation performance in terms of efficiency² and effectiveness³. On the other hand, an application context has a direct impact on applications and implementers and, therefore, is certainly a part of an implantation context. The inclusion of application contexts suggests that boosting portability of an information model shall not jeopardize its extensibility.

Enabling high extensibility and high portability at the same time, Kjellberg, et al. (2009) proposed a two-layer modeling approach composed of a generic stable core model and a specialized ontology. This study concretizes the idea of the two-layer approach as the concept of the implementation model⁴ (Section 3.4). An implementation model encapsulates and compensates data schemas to satisfy requirements in an implementation context. As a new layer in the modeling architectures, the implementation model is a usable abstraction of useful information for implementers. The abstraction means simplifying the complex modeling architecture, hiding what is unnecessary for implementation, and detailing pragmatic support.

With the proposed concepts, i.e. implementation contexts and implementation models, the first part of this study can be summarized as 1) using implementation contexts to accurately understand expectation of implementers on modeling architectures during integration implementation

¹ Implementation context: An environment where implementers use an information model to implement an application (more details in Section 3.3).

² Efficiency: "Resources expended in relation to the accuracy and completeness with which users achieve goals" (ISO, 2010b).

³ Effectiveness: "Accuracy and completeness with which users achieve specified goals" (ISO, 2010b).

⁴ Implementation model: An information model that encapsulates one or more data schemas, with additional supportive functions for data set manipulation, to satisfy information requirements in an implementation context (more details in Section 3.4).

(Section 3.3), 2) developing a methodology to specify implementation models to satisfy information requirements based on implementation contexts (Section 3.4) and 3) enhancing modeling architecture to facilitate development and use of implementation models in system integration (Section 3.5).

Based on the previous introduction, a research question and a corresponding hypothesis highlight the current challenges and the possible resolutions tested by this study:

RESEARCH QUESTION I

How can system integration be enhanced with information models?

HYPOTHESIS I

Introducing implementation contexts and implementation models in a modeling architecture improves efficiency of system integration.

Improvement of efficiency in implementing system integration is the core motivator of this research. For this purpose, an implementation context is firstly defined as an environment where an information model is implemented (Figure 1.6). Efficiency in an implementation context, rather than an application context, determines usefulness of an information model. Thereby, a modeling architecture should be developed to satisfy the implementation context where it is used.

The introduction of implementation contexts suggests a missing link in the development of the existing standardized generic information models. Hence, to interface the existing ones, an implementation model (Figure 1.6) is defined as an implementation-ready portable information model. For implementers, an implementation model acts as a bridge between their programs and data sets.

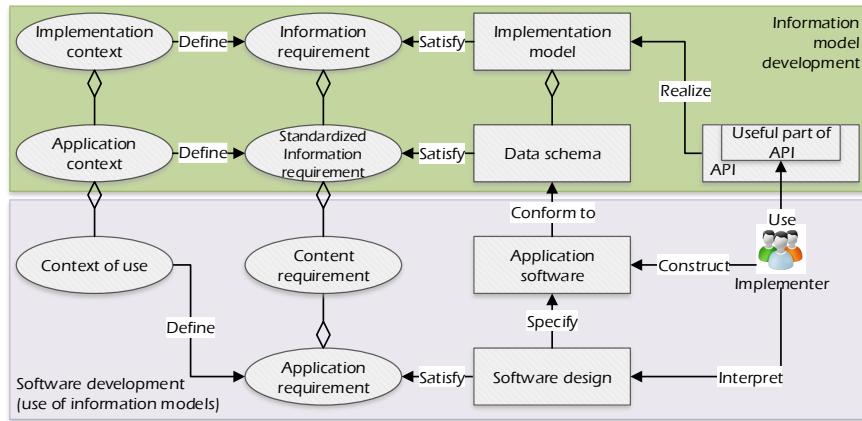


Figure 1.6 An envisioned situation of model development and software development.

A typical use scenario is when an application software should conform to one or more standardized generic information models, possibly tailored. Encapsulating these information models, an implementation model can be realized as an API (Application Programming Interface) directly applicable for constructing such application software. An API is the most efficient and understandable way for implementers. To produce the API, the goal of the proposed modeling architecture will be as guidance to develop and to use an implementation model in a designated implementation context.

To develop and use the API, practical case studies are described in the appended publications A, B, C and E which propose implementation models for different purposes at suitable levels of abstraction. They are also validations of the modeling architecture in various instances of implementation contexts. As a summary paper, the appended paper D describes a Java implementation of the modeling architecture for efficient model driven system integration. The implementation in paper D is a basis for demonstrations in other publications (Figure 1.7). Paper D also makes an architectural comparison between the new solution and the original.

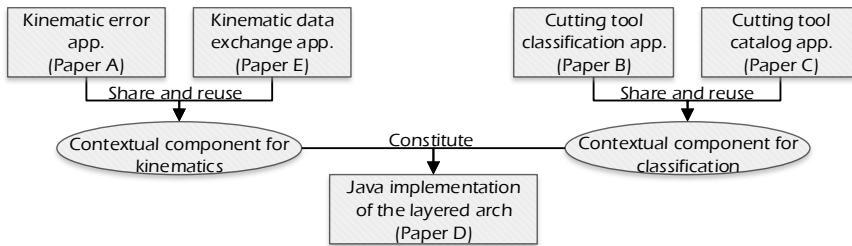


Figure 1.7 Publications relative to the layered architecture.

Within the publications, it can be observed that implementation models are more contextual, with less semantic details and more pragmatic details. Pragmatically, more implementation aspects in an implementation context make the implementation models less complicated and easier to use. The implementation models realized as APIs are constituted by components that are contextual for different system integration domains, e.g. kinematics and classification (Figure 1.7).

This contextualization does not jeopardize the reusability of the information models. The contextual components are harmonized to be reusable among applications in the broad application context designated by STEP. The harmonization has been validated in cases of kinematic modeling in paper A and E and also in cases of classification modeling in paper B and C (Figure 1.7). The paper A on kinematic errors is an extension to general kinematic information representation which is formulated in the paper E. The paper B creates a mechanism for cutting tool classification modeling. Then, the paper C describes a scenario to reuse the classification mechanism for catalogue information communication. In addition, all these use cases are harmonized with the product assembly modeling support as a part of the Paper D.

In conclusion, the first part (Chapter 3) of this study contributes to improving implementation efficiency with a new modeling architecture. To begin with, the causes of implementation efficiency issues should be discovered for system integration based on information models: 1) Development of data schemas is initialized by application contexts not including implementers; 2) in an application context, a large scope and universal acceptance at a semantic level require a complicated modeling mechanism;

3) changeable application contexts require highly extensible modeling architectures that is development-oriented but not friendly for implementers. Clearly, the use of application contexts is a weak link of conventional modeling architectures to support implementation. Based on implementation contexts to compensate the application contexts, a new modeling architecture can lead to information models satisfying needs of implementers. This solution forms the basis of implementing model driven system integration.

1.3 An integrated implementation process

Successfully implementing model driven system integration is not just about programming with information models as enabling infrastructures. Traditionally for implementing system integration, standards with different focuses are treated as technical constraints. A portable solution to use information models is not always available to be directly fed to programming. Programmers are faced with manual interpretation (Section 2.2.2) of sophisticated data schemas that are usually designed for computer interpretation. To resolve this situation, this study aims at guiding use (Section 1.3.2) of information models based on conventional software engineering lifecycle. It requires exploiting potential (Section 1.3.1) of information models in integration architectures and information architectures for system integration.

1.3.1 Two types of unused potential of information models

One type of potential of information models (e.g. ISO 10303) is embodied in highly comprehensive semantic representation for large scope, i.e. rich semantics in practice. The rich semantics leads to complex modeling architectures which have three results for different stakeholders (Figure 1.8): 1) It facilitates qualified model development and 2) formal computer interpretation; 3) it also result in heavy workloads for the direct end-users, the implementers. For implementers, they need smartly use, not suffer from, the complexity in software engineering activities, i.e. implementation.

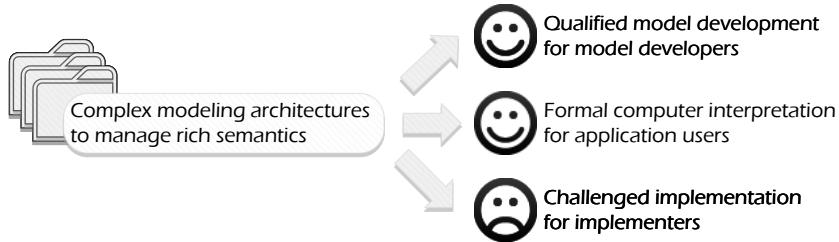


Figure 1.8 Results of the complex modeling architectures

This study claims that the use of an information model can be extended beyond what it is designed for, i.e. as a technical constraint (Section 4.1.1) to facilitate semantic interoperability. An information model has the potential of a reliable knowledge base underpinning an implementation process and facilitating pragmatic interoperability. Semantic comprehensiveness of an information model could help software design by saving effort at specifying domain specific functionalities and content. Thus, software engineers can focus on designing a system to present the functionalities and content in a user-interpretable way, rather than struggling with domain-specific semantics.

The other type of potential of information models is a similarity in procedure between information model mapping and software implementation. The procedural similarity is illustrated in Figure 1.9. Intended use of the information models, as a model mapping process, is similar to an implementation process integrating concern about HCD (Human-Centered Design) where research of contexts is highly valued for system design. (See more about HCD in an engineering context in the appended paper F.) Both processes transfer the same set of domain knowledge from contexts to deliverables through documented artifacts.

The procedural similarity can be utilized at different stages of implementation if a similarity in abstraction levels is taken care of. This potential will be discussed in Section 4.1.3 in depth, based on a context of general product realization. The detailed integration (Section 4.4) will be based on discussion of general stages (Section 4.2) and modeling activities (Section 4.3) of software implementation.

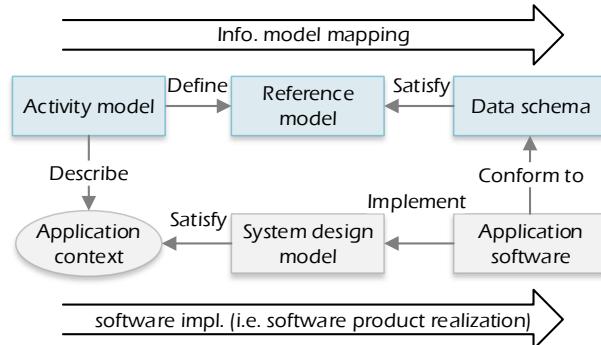


Figure 1.9 Software implementation utilizes different types of information models in a modeling architecture.

1.3.2 Make use of the potential in process integration

This integrated process is certainly a new way to perform implementation in an interdisciplinary context of model driven system integration. In Section 4.5 through Section 4.9, five stages with eleven activities are introduced to formulate the interdisciplinary process. Figure 1.10 illustrates the five stages of the new process in its diagonal, which displays the stages' relations to the implementation activities and the information mapping.

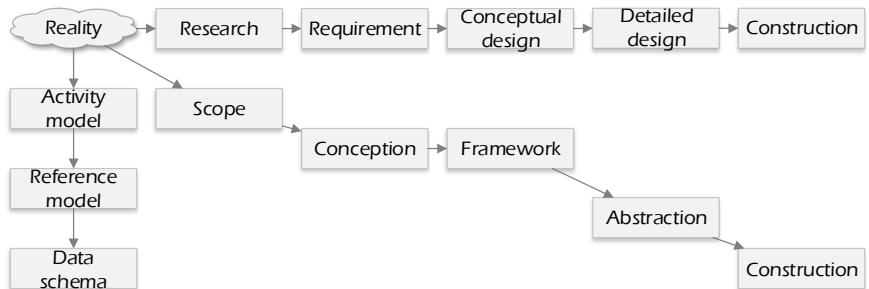


Figure 1.10 An integrated process for model driven system integration.

Note that human-model interaction and communication is a significant feature in the application context of engineering activities (Hedlind, 2013). In this context, system integration is frequently built upon systems characterized by intensive human-computer interactions. The paradigm of

HCD is adopted in this study to address design concern for interactive systems¹ (Section 4.3.1). In particular, presentation of content should be architected to ensure best human interpretability, findability and ease of navigation (more about information architectures in Section 2.5). Integration of all the subjects is a challenge for implementers who are often domain experts for system integration in practice (Section 4.3.2).

To support these implementers, the focus of the second research question is integration of the processes for model driven system integration. The processes to be integrated include the information model mapping, the conventional software engineering lifecycle models and the HCD. This integrated process should be model driven: Standardized information models, as a reliable resource for domain knowledge, are used to effectively underpin the software engineering activities.

RESEARCH QUESTION II

Where and how can information models be used to support system integration?

HYPOTHESIS II

Facilitated with existing modeling architectures and HCD (Human-Centered Design) principles, a software implementation process enables effective use of information models for system integration.

This study aims at a novel way to use the support from information models in system integration. Based on conventions in software engineering and HCD, stage-by-stage integration of the information models in implementation is resolved in Chapter 4. With this integration, implementers actively use information models in the suitable stages of the conventional processes, e.g. requirement engineering, architecture design and interaction design.

¹ Interactive system: "Combination of hardware, software and/or services that receives input from, and communicates output to, users" (ISO, 2010b).

1.4 Main contributions

1.4.1 Identification of the root cause of challenges

The challenge lies in a mismatch between existing modeling architectures and the implementer needs. Implementation is the central activity of using the information models and is supported by modeling architectures. However, when modeling architectures are defined, needs of implementation are not captured. In particular, the data schemas are not sufficient to support implementation practices, and the designated ways to use an information model cannot support implementation efficiently.

Where does this problem originate from? Modeling architectures are mainly driven by how well a data schema can be developed (more details in Section 2.2), rather than by how well a data schema can be used. Standardized information models, e.g. ISO 10303 STEP, have prepared rich semantics in a broad scope for applications. Reasonably, model developers design a relatively complex modeling architecture to facilitate comprehensive standardization of rich semantics in a broad scope and an extensible fashion.

In this respect, benefits easily become weakness. This complexity could easily hinder pragmatic use of an information model, resulting in low portability and low implementability (more details in Section 3.2). For instance, several generations of the ISO 10303 modeling architectures have adopted more and more sophisticated multiple-layer structures to support development of comprehensive data schemas. The structures strive to ensure validity, completeness and extensibility of semantic representation in a data schema. However, the complex structures result in implementation difficulties such as instantiation inconsistency (von Euler-Chelpin, 2008). Without proper guidance (von Euler-Chelpin, 2008; Kjellberg, 2009), these structures could be a hurdle for implementation.

Figure 1.11 exemplifies a traditional modeling architecture of ISO 10303. As illustrated, model developers follow the upper process (development of a data schema) to specify a qualified data schema by creating a series of model artifacts. To use the data schema during a specific implementation project, a segment of the data schema will be located by implementers through the lower process (use of a data schema) in Figure 1.11.

This lower process could demand implementers to process plenty of documents (e.g. Chapter 2 of Paper D) to understand e.g. relevant content specified in a modeling architecture.

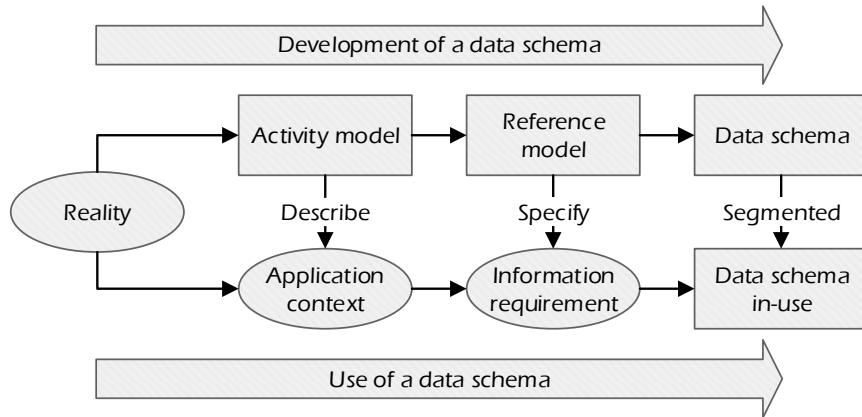


Figure 1.11 The conventional modeling architecture of ISO 10303 STEP, based on textual description by Anderl and Wasmer (1997).

This modeling architecture (Figure 1.11) could cause technical problems and business problems. The technical difficulty is easily observed when applying such a modeling architecture in implementation. The modeling architecture leads to ambiguity of task allocation between roles of implementers. Practically, it is better if application designers perform the model mapping as exemplified in Figure 1.11 because they know the contexts and the requirements better. However, this is not intuitive (more details in Section 3.2) because, as a routine to use any standard, programmers should interpret and apply the standards as constraints in programming.

This mismatch between the modeling architecture and implementation needs could result in business issues on implementation. Firstly, it makes existing inadequate interoperability for product realization worse, by hindering efficient use of standardized information models. Secondly, the use of information models cannot effectively guarantee interoperability, in that conformity is totally determined by implementers who likely lack experiences with the modeling architectures; it can lead to incompatibility when implementations are based on incomplete assumptions. Thirdly, it is

a waste of resources that not many engineers can reap great benefits of the standardized information models as a comprehensive knowledge base.

1.4.2 Solutions

To tackle this situation, an information model can be either enhanced (research question I), or be used in an alternative way (research question II). The solutions aim at resolving the challenges by redesigning modeling architectures (Chapter 3) and the process (Chapter 4) to use the information models in system integration. Two strategies are proposed to take on the challenges for two aspects of modeling, development (Chapter 3) and use (Chapter 4). That is, developing information models should take care of its designated use scenarios, namely the implementation practices within the software lifecycle (Chapter 3). Use of information models should be aligned with software engineering activities from which integration architectures and information architectures are core deliveries (Chapter 4).

The two strategies form the backbone of this thesis: 1) Chapter 3 proposes a modeling architecture to guide a new way to develop an implementation-friendly information model. In the modeling architecture, concepts of implementation contexts and implementation models are introduced to satisfy the needs of implementers. 2) Chapter 4 designs a process to implement system integration that can use information models in an effective fashion. The purposes of the process include exploiting potential of existing standardized information models and speeding up and simplifying the implementation activities.

In addition to those two major chapters, the rest of the thesis fills in the background, validation and conclusions (Figure 1.12). This chapter briefly introduces research questions that are answered by the two chapters, in relation to hypotheses that shall be verified. Chapter 2 introduces the scope of study and reviews relevant topics. Chapter 5 describes several case studies that implement and validate the methods developed in Chapter 3 and Chapter 4. Chapter 6 concludes this study, with discussion of contributions, limitations and suggestions of future work.

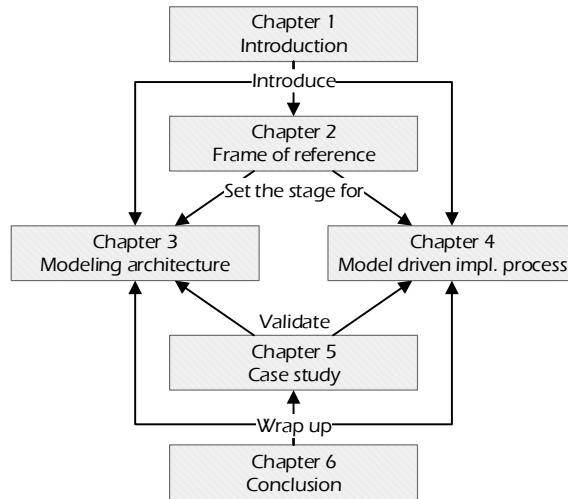


Figure 1.12 Thesis structure.

1.4.3 New concepts

This study creates two new concepts to support development of modeling architectures. At first, this study defines an implementation context (more details in Section 3.3) as a primary environment where an information model is used. The implementation context plays an important role to understand what an information model is needed. With an implementation context, a modeling architecture provides formal representations of information suitable for implementing applications using an information model. On the other hand, an application context of an information model only lets model developers know representation suitable for application use.

This study also proposes an implementation model (more details in Section 3.4), interfacing data schemas, to facilitate efficient software construction based on the implementation context. The implementation context guides development of the implementation models as a new layer. A modeling architecture is developed to guide how to use the implementation models for model driven system integration.

The most important inter-related components in an implementation context are illustrated in Figure 1.13. An application operates on the information models and is interacted with by users in a context of use. If necessary, an application should be able to pool the information models with data from different domains to perform different tasks. Implementers perceive all the interactive needs as a context of use and, accordingly, find related information models to develop the application. Information models will be used by users, applications, and implementers at different levels of abstraction. That is, users possess data sets, applications operate on data schemas to process data sets, and implementers use implementation models to interface the data schemas. Note that these roles may overlap on particular persons in an implementation context (Section 3.3). End-user programmers are typical examples of this scenario, where an implementer of an application is also an end user of the application.

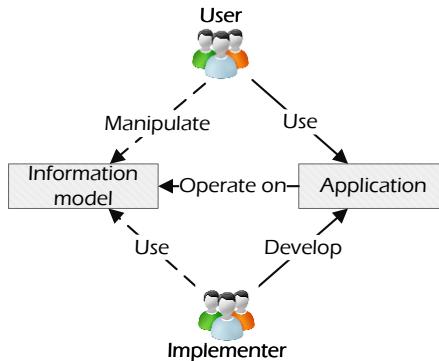


Figure 1.13 An application as an intermediate between stakeholders and information models.

Chapter 2

Frame of reference

Art and science have their meeting point in method.
Edward Bulwer-Lytton

Model driven system integration for product realization is a highly interdisciplinary topic (Section 1.1.1). Based on knowledge in different disciplines, this chapter sets the stage for this study. Interoperability (Section 2.1) is the most fundamental quality requirement of system integration. Standardized information models (Section 2.2) facilitate interoperability in system integration for product realization. Besides, the standardized generic information models can be treated as a reliable knowledge base for function design and content design in system integration implementation. In general, models are important infrastructures to maximize usability¹ and reusability of information in engineering activities (Section 2.3). Implementation of system integration requires plenty of decisions about high-level structural designs; this leads to discussion about architecting activities (Section 2.4). Engineering applications are usually HCI (Human-Computer Interaction)-intensive and content-rich. This fact demands interaction design (IxD) at an architectural level adopting principles of information architecture (IA, Section 2.5).

2.1 Interoperability

This section will begin with description of the challenge (Section 2.1.1) that is represented by a lack of interoperability. It is followed by a discussion of previous frameworks to define interoperability as a quality requirement (Section 2.1.2). The discussed frameworks are noticeably originated from semiotics (Section 2.1.3). In the end, the pragmatic aspect is stressed as the focus of this study (Section 2.1.4).

¹ Usability: “Extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use”. (ISO, 1998b)

2.1.1 Interoperation as a challenge

The landscape of manufacturing industry has been significantly changed by Information Technology (IT). Competitiveness of IT enterprises is expressed by heterogeneity in technical infrastructures, operational practices and business strategies. The heterogeneity is particularly reflected in Computer aided technology (CAx) that has been changing the landscape of engineering activities in manufacturing industry. The industry demands CAx vendors to introduce more and more CAx functionalities for different business requirements based on different product lifecycle stages. However, no single IT product can satisfy all the requirements. Product realization heavily relies on interoperation of multiple types of applications. In addition, engineering activities frequently require experts working in a distributed manner (Wang, et al., 2002). Hence, for the interoperation, effective information sharing and exchange is needed.

Nonetheless, the heterogeneity could easily result in isolated systems with incompatible data. Moreover, continuous changes everywhere in industry increase discomfort of cross- and intra-enterprise communication and collaboration. Engineers are frequently challenged by unresolved interoperation issues with existing systems (ex-post compatibility) and new systems (ex-ante compatibility, David and Greenstein, 1990). This challenge was pointed out by Shannon (1948) in the dawn of the Information Age: “The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point. Frequently the messages have meaning”.

Especially, what is communicated is with product data which can be created, processed and stored in a fragmented manner by disparate applications. The fragmented manner is exhibited by mismatches in syntax, semantics, or pragmatics (more details in Section 2.1.3). Fixing these mismatches by different strategies is the most important work to enhance interoperability.

Interoperation based on product data is required for both communicating existing systems and delivering new CAx functions to support professional engineering tasks. For such professional activities, implementing any functionality means dealing with rich content, e.g. models of geometry, properties and features. When content is available in existing systems, it is

not value-adding either to recreate a data schema from scratch or to ask users to recreate data sets from scratch. Hence, properly reusing those existing content is a key for efficient implementation. For instance, (Figure 2.1) implementers should consider to reuse the existing data schema when integrating a new function (e.g. kinematic modeling).

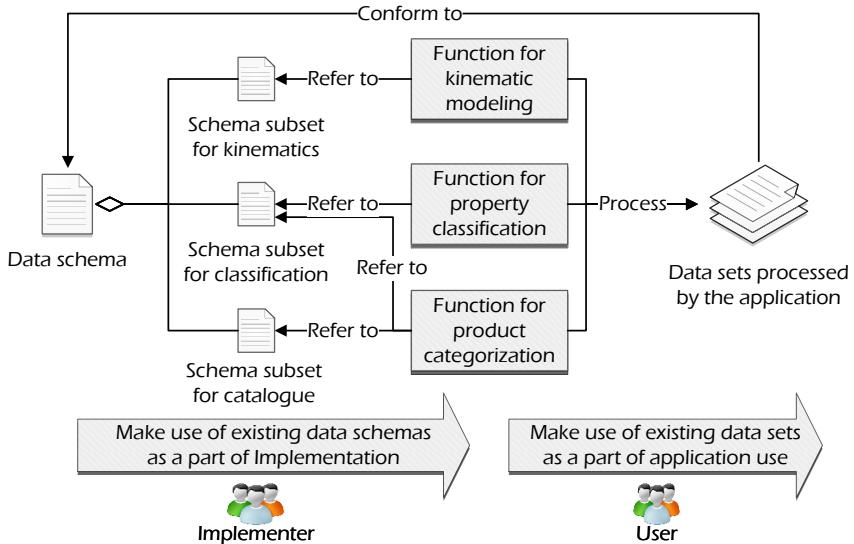


Figure 2.1 Reuse of existing data by implementers and users.

The above case particularly stresses that reuse of existing data facilitates efficiency of implementation. Implementers should concentrate on reuse of data schemas to enable application users to reuse data sets. Implementers have much work to do for successful reuse (more details about how to implement a data schema in Section 2.2.2). Implementation could be troublesome if there is no clear guidance toward the implementers. Hence, the implementers, as data schemas users, demands sufficient support to use the data schemas. Why and how the existing support is insufficient will be discussed in Section 1.2 (in brief) and Section 3.2.

Failing to reuse product data could result in a tool-driven manner which is noticeable in practical engineering activities. The *tool-driven* manner is a problematic way to complete engineering goals greatly depending on available tools rather than engineering specifications. It results

in limited choices of personnel, limited choice of tasks, limited ways to perform tasks and limited information at hand. These are all determined by ease to use necessary product data by applications suitable for the desired goals. These barriers to reuse the product data force engineers to settle for what are provided by proprietary tools. Otherwise, engineers have to manually process or re-type the data, when faced with non-reusable data sets or non-useable interfaces. Richard Stallman vividly illustrated this manner in the scope of application software several times in other contexts (TED, 2014, Figure 2.2).

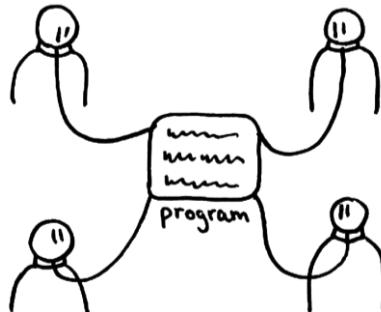


Figure 2.2 A situation where programs (i.e. application software) control users, presented by Richard Stallman at TED (2014)

2.1.2 How to understand interoperability

These challenges make interoperability a critical quality requirement when developing computerized information systems. The *interoperability* is “the ability of two or more systems or components to exchange information and to use the information that has been exchanged” (ISO/IEC/IEEE, 2010). It is also a critical quality required to develop SoS (System of Systems, DoD, 2008). It relies on complicated integration solution based on a group of involved components and structures, e.g. interfaces, communicating entities, contexts, data, services and standards. This is a quality increasingly demanded in almost every domain related with IT, yet difficult to achieve appropriately (Brownsword, et al., 2004).

Previous researchers have defined different frameworks to depict typography of this area, for instance, Martin (2005) put forth a three-level framework to classify interaction manners between systems:

- Unified systems: Interacting via same conceptual representation.
- Integrated systems: Interacting via agreed fixed representation.
- Interoperable systems: Interacting via dynamic interaction rules.

It is noticeable that integration and interoperation are similar in many contexts. Brownsword, et al. (2004) suggested that both terms can be used interchangeably, while integration may imply more tightly coupling, as what Martin (2005) pointed out in his framework.

There are a number of aspects that determine interoperability. Messages are the core elements to characterize interoperability at three levels (Pokraev, et al. 2007), which can be described with more concrete concepts:

- Syntactic interoperability: Formats are compatible.
- Semantic interoperability: Meanings of messages are shared.
- Pragmatic interoperability: Expectation of effects of messages are shared.

Based on this work with extension, Sharif (2009) developed a similar four-stage framework to describe system integration, with emphasis on inter-level dependency. The stated here were suggested to be implemented sequentially toward best performance of system integration:

- Interconnectivity: Telecommunication infrastructures are facilitated.
- Functional integration: Networks, protocols, formats and procedures are defined technically for integration.
- Semantic integration: Consistent semantics is achieved.
- Optimization & innovation: Integration enables improvement of technology or business processes.

2.1.3 Semiotic aspects of interoperability

Obviously, the frameworks to define interoperability are influenced by basic semiotic concepts. Scope of semiotics can be defined as a famous trichotomy, syntax (or syntactics), semantics and pragmatics. Morris (1938) coined this trichotomy and identified it as three branches of semiosis. The other division made by Morris (1938) was so-called pure semiotic and de-

scriptive semiotic. The two division systems form a structured view to describe the scope of semiotics. The trichotomy was considered to be inherited (Posner, 1985; Nöth, 1990) from the medieval trivium (grammar, dialectic and rhetoric, Greene, et al., 2012) and Peirce's triadic model of the sign (representamen, object and interpretant, Nöth, 1990).

Although validity of this trichotomy has been questioned since its introduction (Lieb, 1971), the triad is still a useful tool to describe a “sign-situation” (Figure 2.3, Morris, 1939), e.g. representation, abstraction, modeling, etc. The triad includes three components of semiosis and three dimensions of semiosis, which lays an important early foundation for research on modeling. Many researchers on information systems have applied, integrated, and extended this triad into layers to discuss relevant concepts.

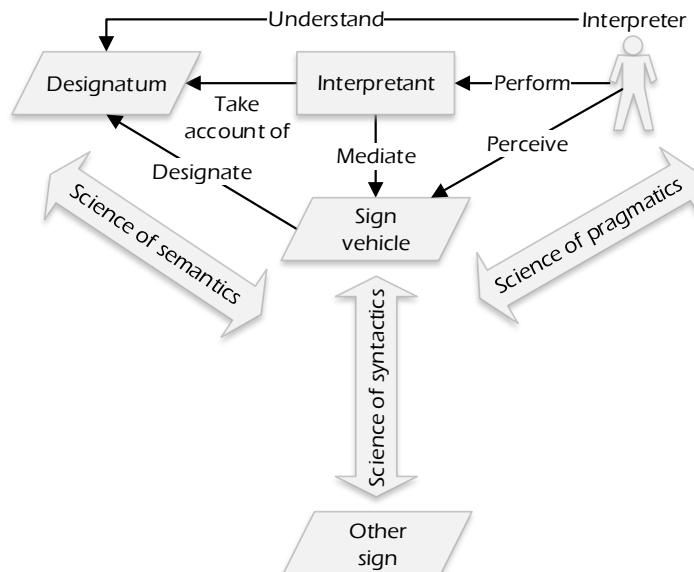


Figure 2.3 “A sign-situation, or a process of semiosis”
(drawing based on Morris, 1939).

Dealing with conceptual modeling for requirement engineering, Lindland, et al. (1994) made a remarkable early contribution to adapt this trichotomy in the modeling domain of software engineering. In their framework, four basic sets were defined for possible statements in a conceptual model (Figure 2.4):

- Language is a set of all statements in a syntax. The syntax including an alphabet (a set of modeling construct) and a grammar (a set of composing rules).
- Domain is a set of all statements to solve a specific problem. It includes necessary knowledge for a specific context, instead of an entire context of a discipline.
- Model is a set of statements, including statements explicitly made and statements implicitly deducted according to the “language”.
- Audience interpretation is a set of statements in the “model” understood by audience, correctly and completely.

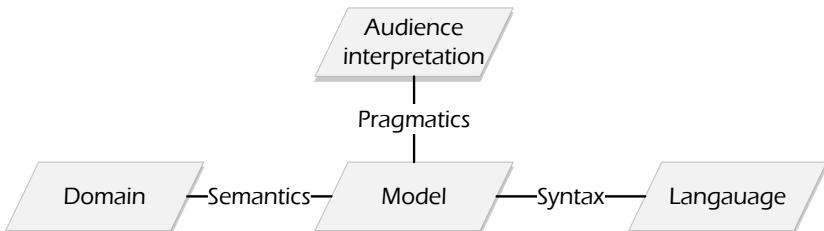


Figure 2.4 Four cornerstones and three connections of the framework proposed by Lindland, et al. (1994)

Model, abstraction and level of abstraction

To put it simply, a *model* is a collection of representations to describe an existing or a future system (Blanchard and Fabrycky, 1990). Minsky (1965) defined a *model* from the perspective of purposes, “To an observer B, an object A* is a model of an object A to the extent that B can use A* to answer questions that interest him about A.” IEEE (1998a) defined a *model* as “a representation of a real-world process, device, or concept”, while the characteristic of the model to suppress certain aspects was emphasized by IEEE (1998b). This study adopts the definition of a *model* concluded by ISO/IEC/IEEE (2010), based on previous ones defined by ISO, IEEE and IEC, as “a semantically closed

abstraction of a system or a complete description of a system from a particular perspective.” In an information model, it is critical that represented concepts and their relationships should be used to specify content which matters most for a subject domain (Hackos, 2002). Of course, there should be rules, constraints and operations to enrich semantics (Lee, 1999). Specifically, this thesis uses the term “model” in three contexts:

1. An information model in the domain of product realization is a formal model to capture a bounded set of information to meet a specific requirement (ISO, 1994). By this definition, all CAx should be developed with information models to some extent. The model driven approach will be defined in this context as well.
2. A system design model in the domain of software engineering is a model to describe contexts, requirements, architectures and designs as deliveries of model based design activities in a software implementation process.
3. A data model as a component of a software architecture. It is an important part in the architecture pattern MVC (Model-View-Controller, more details in Section 4.8 and 4.9).

ISO/IEC/IEEE (2010) took a model as a kind of an abstraction and it also defined an *abstraction* as “a view of an object that focuses on the information relevant to a particular purpose and ignores the remainder of the information.” Abstraction as an action indicated a selection process (Ross, et al., 1975), i.e. to extract relevant information but to omit irrelevant details. In conclusion, as an entity, an abstraction denotes a model describing a system for a purpose; as a procedure, an abstraction uses *levels of abstraction* to construct “a view of an object at a specific level of detail.” (ISO/IEC/IEEE, 2010) At a higher level of abstraction, more important information is emphasized and typically less details are revealed.

In this framework of Lindland, et al. (1994), model developers were supposed to be concerned with appropriateness between a “model” set and the other three sets. These concerns were addressed by the three qualities (Figure 2.4):

- Syntactic quality: “How well the model corresponds to the language”.

- Semantic quality: “How well the model corresponds to the domain”.
- Pragmatic quality: “How well the model corresponds to its audience interpretation”.

For each quality, practical issues in terms of goals, means, validity, completeness and feasibility were represented by set theory notations. With this solution, Lindland et al. (1994) offered a systematic way to organize and utilize those principles for modeling in software engineering. Their framework aims at a deeper understanding of modeling qualities than previous attempts consisting of just unorganized quality properties. Until now, researchers on conceptual models or software architectures are still producing these “bread-and-butter collections”, e.g. annotated, appropriated, clean, formal, modifiable, traceable and verifiable.

Hofmann (2003) extended this use of Morris’ trichotomy to integration of model based computerized information systems in the domain of modeling and simulation. In his study, models were no longer statements used for conception, requirement analysis and architectural design. Instead, the models were used to manage “tremendous complexity” of reality. In other words, the models were regarded to be a substitute for designing or controlling state transition of a real system (Figure 2.5). In this illustration, real dynamics of a system β^S was too complex to be directly understood and managed. A model was supposed to be developed with necessary abstraction, idealization and simplification during φ (Figure 2.5). Model dynamics, β^M , described or predicted the system dynamics. After execution of model dynamics, a computerized information system could “re-translate” the new model state to a new system state, in ψ .

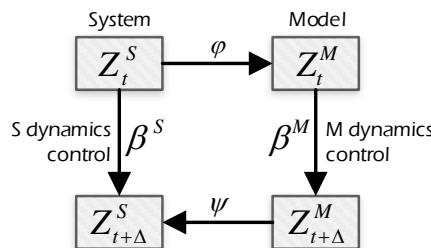


Figure 2.5 Model based computerized information systems (reproduction based on Hofmann, 2003).

Integration of model-based computerized information systems was all about harmonizing different models, so that content and functionalities in different systems could be shared and reused. Technics, syntax, semantics and pragmatics were identified as essential preconditions concerning model development for system integration:

- Technical aspects were realized by selecting formal protocols for networking and interfacing. Besides basic coupling functionality, a feasible protocol should fulfill needs in flexibility, maintainability and additional services (including authorization, time management, security, etc.)
- Syntactic aspects required “automatically processible data” in a formal modeling language. The modeling language could be challenged by 1) a lack of personal learning incentives, 2) a lack of expressiveness and 3) difficulties of understanding.
- Semantic aspects required “a predefined ontology” as a glossary to attribute meanings to syntax. Besides, semantics for model dynamics (β^M) was difficult to integrate only with a static standardized ontology. Hence, Standardized algorithms for elementary processes in the domain were also required.
- Pragmatic aspects required model dynamics (β^M) in a consistent manner after the integration. It led to standardized β^M . However, it was always difficult to harmonize context, broadness and resolution of models even with same syntax and semantics. A less compulsory solution is to standardize graphical user interfaces (GUI), although it is not realistic.

Influenced by these preconditions, Tolk (2004) proposed his framework to describe levels of conceptual interoperability, where data is considered as the centric attribute in these five levels of system interoperability (Tolk, 2003). The target of Tolk (2006) was also to maximize interoperability at a pragmatic level. The framework is called LCIM (Level of Conceptual Interoperability Model), of which the first version is composed of six levels:

- No connection at all.
- The technical level: Physical connectivity is established.

- The syntactical level: Standardized formats and protocols are established for data exchange.
- The semantic level: Common reference models are established for information (data + contexts) exchange.
- The pragmatic/dynamical level: An unambiguous form is established for knowledge (use and applicability of information) exchange.
- The conceptual level: A common view of the world (epistemology) is established for theory of domain knowledge (e.g. reference to its limits and validity) exchange.

2.1.4 Pragmatics: A focus of this study

Synthesizing the above categorization theories as well as insights gained from project experiences, this thesis selects three most important and feasible levels of interoperability: Syntax, semantics and pragmatics (Figure 2.6). A higher level indicates easier integration (Tolk and Muguirea, 2003). The pragmatic level of interoperability is the uppermost need regarding information for implementation of interactive system in practice. Note that the pragmatic interoperability in this study focuses on a technical level rather than a business level (Asuncion and Van Sinderen, 2010). Namely, it requires mutual understanding on data intention (expected effects), data use (actual effects) and a context where communication happens (Liu, 2009). The modeling architecture (Chapter 3) reaches this understanding by employment of the implementation models based on the implementation contexts.

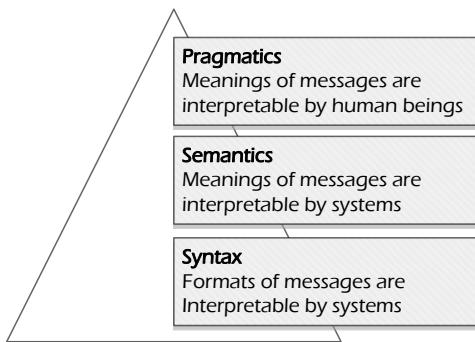


Figure 2.6 Three levels of interoperability concerning this study.

The levels of semantics and syntax are foundations for the pragmatic level. In general, consistent syntax and semantics can assure acceptable interoperability for many cases without human involved. As illustrated in the upper part of Figure 2.7, meanings (e.g. product information) in communicated data sets are interpretable by computers. Afterwards, the computers decide how to present the meanings to application users and what kinds of manipulation on the meanings can be done by the application users. To an extent, this is helpful for the application users, as computers are much better to store information massively, access information in a focused way, and to reason beyond human cognitive ability (Arp, et al., 2015).

However, until now, to make the semantic interoperability a reality, human interpretation is inevitable in that human implementers create all the functions to process the semantics. When the meaning in data sets is automatically presented to users, implementers have to find a way to manually interpret the meaning of the data schemas (the lower part of Figure 2.7). Taking implementers into account, performance of human interpretation of data schemas is represented by pragmatic interoperability.

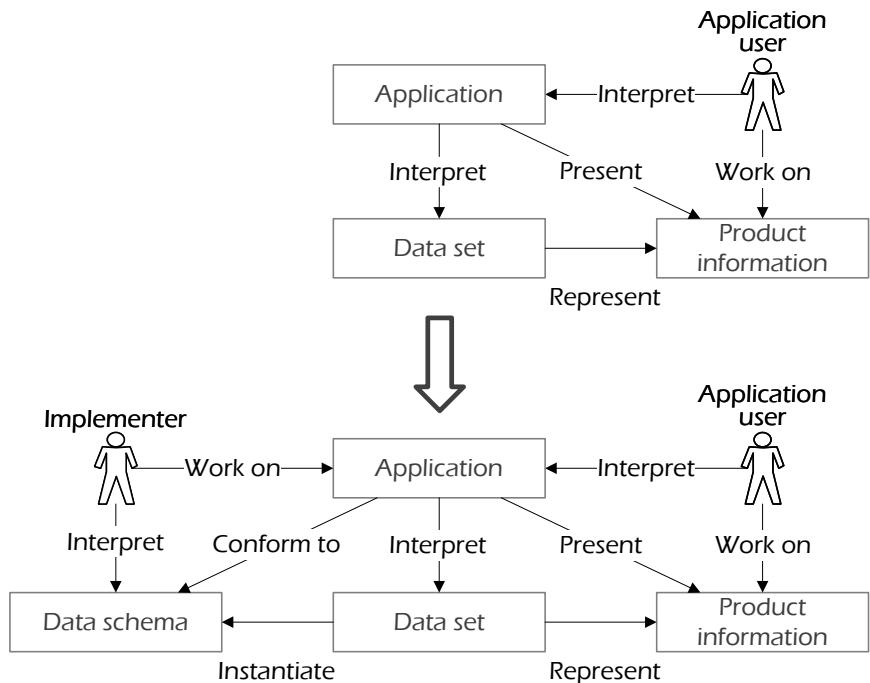


Figure 2.7 Computer interpretation of information models (data sets) needs human interpretation of information models (data schemas).

This study uses the three-level framework in two ways. Firstly, in Chapter 3, a modeling architecture is proposed to facilitate pragmatic interoperability. A missing link in development of information models is identified and resolved in a context of model driven system integration. It corresponds to Hypothesis I. Secondly, Chapter 4 takes advantage of the high semantic interoperability of standardized information models to underpin an implementation process (Hypothesis II). This measure is critical for implementers because of the interdisciplinary nature of the implementation activities.

2.2 Standardized information models

The system integration requires qualified information models. Standardized protocols determine success in interoperability and integration (Martin, 2005). Architecting system integration means exploring a best way to navigate data through the protocols. There are numerical styles of the navigation: Shared database, file transferring, remote procedure invocation, message bus, etc. (Hohpe and Woolf, 2003) No matter which style is chosen, an information model has to be used to represent information to be exchanged. Hence, success of system integration depends on not only interoperability of applications, but also interoperability of information models.

In particular, important facilitators for system integration are standardized generic information models, of which specifications are known as generic information standards. A major capability of such standards is the interoperability for communicating meaningful information (Kazman, 2014). The standards are also known with a broad scope and wide applicability which can be expressed by the generic PPR (Product-Process-Resource) framework proposed by Nielsen (2003). Examples of the generic information standards are AP214 (ISO, 2010a) and AP242 (ISO, 2014b) which form the base of development, use and discussion of modeling architectures in this thesis. Extensive discussions about benefits of the generic information standards have been made by von Euler-Chelpin (2008) and Kjellberg et al. (2009). In brief, benefits of the standardized generic information standards include:

- Qualified semantic representation,
- Reliable methodologies for development and use,
- High extensibility.

This section will introduce key concepts for information modeling at first, e.g. application contexts, information requirements, data schemas and data sets. These concepts are critical for specifying an information model in a modeling architecture.

Information model, data schema and data set

An *information model*, as a central concept for model driven system integration, was defined as a “formal model of a bounded set of facts, concepts or instructions to meet a specific requirement.” (ISO, 1994) An information

model can be composed by an interrelated or integrated set of information models (Schenck and Wilson, 1994), particularly in forms of widely accepted information standards, e.g. ISO 10303. For instance, the data schema of ISO 10303-214, integrating several information models, is a widely used standardized generic information model. This study only focuses on the standardization effort when discussing information models in general, as the need of system integration.

There are many ways or criteria to categorize information models. This study is concerned about one way to categorize information models, according to levels of abstraction of represented information, into two categories, data schemas and data sets. A *data schema* is an information model to represent types of information in a particular context. This is a definition combining an old definition of “information model” by Schenck and Wilson (1994) and a more specific definition of “application schema” by ISO (1998a). With the definition of a data schema, a *data set* is an information model instantiating what defined in a data schema. A data schema formally specifies conventions in syntax and semantics to construct a data set in a computer interpretable way. According to the definitions of the information model and the model, both the data schema and the data set are supposed to represent a system to meet a specific requirement. The system described by a data schema has a larger scope and a higher level of abstraction than the one described by a data set.

2.2.1 Application contexts and information requirements

Design of an application shall begin with understanding the context where the system is used, so shall design of an information model. It is always easy to draw fascinating graphical interfaces, to invent tempting functions and to implement complicated algorithms. However, it is the context that justifies whether the developed systems are useful or not. Development projects without adequate analysis on contexts of use can end up with systems dissatisfying user needs (Bevan, 2013).

Software implementers took decades to realize that their deliveries should be human-centered, rather than programmer-centered (Grudin, 1990). Human-Centred Design (HCD, ISO, 2010b) starts from understanding contexts of use. Its importance was confirmed by famous IT system design methodologies such as scenario-based design (Rosson and Carroll, 2009) and goal-oriented requirement engineering (Van Lamsweerde,

2001). The contexts help implementers to obtain complete, pertinent, traceable, readable, stakeholder-oriented, conflict-free requirements (Van Lamsweerde, 2001) to be implemented.

To understand the requirements of what should be represented, ISO 10303 devised the concept “application context” (ISO, 1994) which is useful for information modeling (development and use/implementation). Figure 2.8 illustrates how an application context is used in software implementation. The application context forms a basis for implementers to produce applications for application users. In an application context, there may be other components than just application users: Tasks, domain knowledge, technological environments, business environments, etc. This idea seems promising also for specification of information models that are obviously in a similar subject domain and a related problem space. As defined in Figure 2.9, an information model is developed based on a designated application context and is intended to be used by a group of processes (an application) situated in the application context. This pattern can be observed in widely used standards such as ISO 10303 STEP.

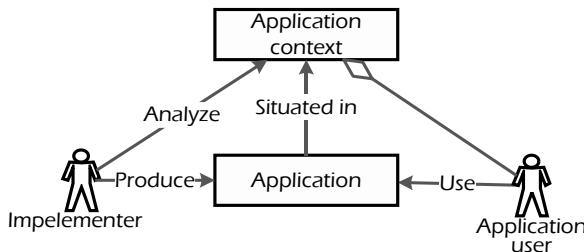


Figure 2.8 Application context used in software implementation.

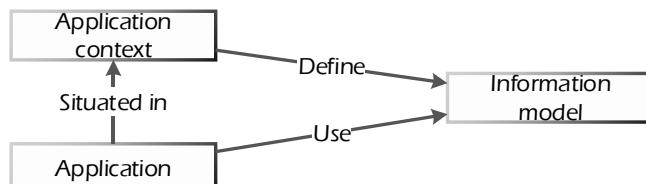


Figure 2.9 The application contexts defining information models.

Requirement, application context, application and information requirement

In the domain of software engineering, a requirement is a “statement which translates or expresses a need and its associated constraints and conditions” in a high-level form (ISO/IEC/IEEE, 2011a). This definition can be applied to a system as a whole to accomplish an objective, or to a software component of a system of which elements include hardware, firmware, people, information, services, etc. (INCOSE, 2010) The requirements represent needs of not only users or operators, but also other types of stakeholders, e.g. customers, regulatory authorities and maintenance men/women.

Software engineers specify *requirements* (namely, intended use of a system) based on an *application context* (ISO, 2003a) which is an environment where data is used in a specific application (a modified definition based on ISO, 1994). Toward common interactive systems, a similar and newer concept is a *context of use*, “users, tasks, equipment (hardware, software and materials) and the physical and social environments in which a product is used.” (ISO, 2010b) This definition used a larger scope with identified components that constituted a context in relation to users for the system-of-interest. A context can be treated as a function of the components. A system is normally designed to work in a range of context of use, i.e. several combinations of the contextual components. A complete description of a designated context is not realistic for any system, because a large number of contextual factors are essentially meaningless for a specific application. Moreover, a system in any stage of its lifecycle (e.g. development, deployment and production) may reshape the context for the system. During a lifecycle of a system, some components may be subject to changes but some may not. System designers need to decide a set of ranged aspects that are relevant for a system-of-interest and their changeability (ISO/IEC, 2014).

As what it was proposed for, this study delimits the concept of the application context only applicable to information models and applications using the information models and that an *application* can be specialized by *application software*, “software that is specific to the solution of an application problem” (ISO/IEC, 2015). In requirement engineering, there should be many types of requirements elicited from a context and information models and applications should be based on different contexts and different consequential sets of requirements. Traditionally, an information model is treated as a technical

part of an application, so an information model can be defined by a subset of requirements necessary for developing an application. The subset is *information requirements* (Figure 2.10), i.e. a set of requirements in an application context that should be satisfied by a data schema. ISO (2016c) defined three types of application contexts that can be used to specify information requirements: 1) Exchange, 2) integration and sharing and 3) long term archiving and retrieval.

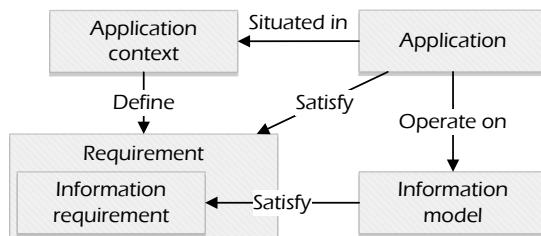


Figure 2.10 A traditional view of information requirements as a subset of requirements to be implemented in information models.

However, a lack of distinction of data sets and data schema makes it ambiguous based on this understanding of the relation between application contexts and information models. A data schema is developed to describe a way to represent information supporting an application context. A data set, (probably partially) instantiating a data schema, is operated by an application to perform some tasks in an application context. For system integration, the applications shall comply with a particular data schema to correctly interpret the corresponding instance data sets from other sources, in order to achieve interoperability. Thus, Figure 2.9 can be extended to Figure 2.11.

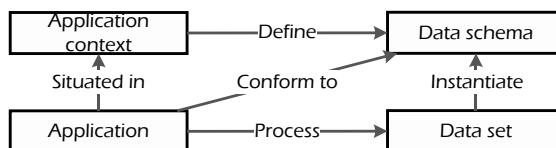


Figure 2.11 A data schema and a data set.

If only considering use of information models in implementation, a more detailed illustration can be obtained in Figure 2.12 combining Figure 2.8 and Figure 2.11. Some new concepts are also introduced: 1) Data sets are representations of product information used by engineers; 2) With the help of data sets, product information can be output and be shared with others, i.e. customers; 3) an application context may include other types of components, e.g. product information and customers; 4) the relation between application contexts and data schemas are omitted, since creation of data schemas is not relevant in implementation. 5) Information models, in the form of data schemas and data sets, are technical constraints for implementation. For instance, a product designer uses a CAD (Computer-Aided Design) system to produce a product design model and then hands the model to a process planner. In this case, the product designer is an application user, the process planner is a customer and the product design model is a form of product information represented in a computer-interpretable data set. Customers do not care about how well an application is developed as long as the product information is well captured for their own tasks. Therefore, the customer is an indirect user (ISO/IEC, 2014) who does not interact with the application directly but is affected by the application's output indirectly.

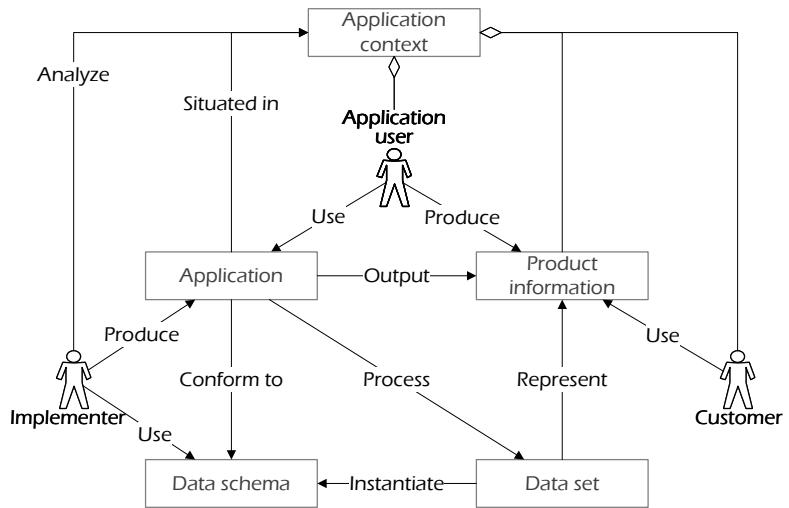


Figure 2.12 Application context, the current basis for application development.

Meanwhile, as illustrated in Figure 2.11, application contexts are also used to develop data schemas. Thus, Figure 2.12 can be extended to Figure 2.13 with additional blue elements. Model developers are introduced in this new illustration to make use of application context and to create new information models. As illustrated, the major output by model developers is data schemas which will be used by implementers directly.

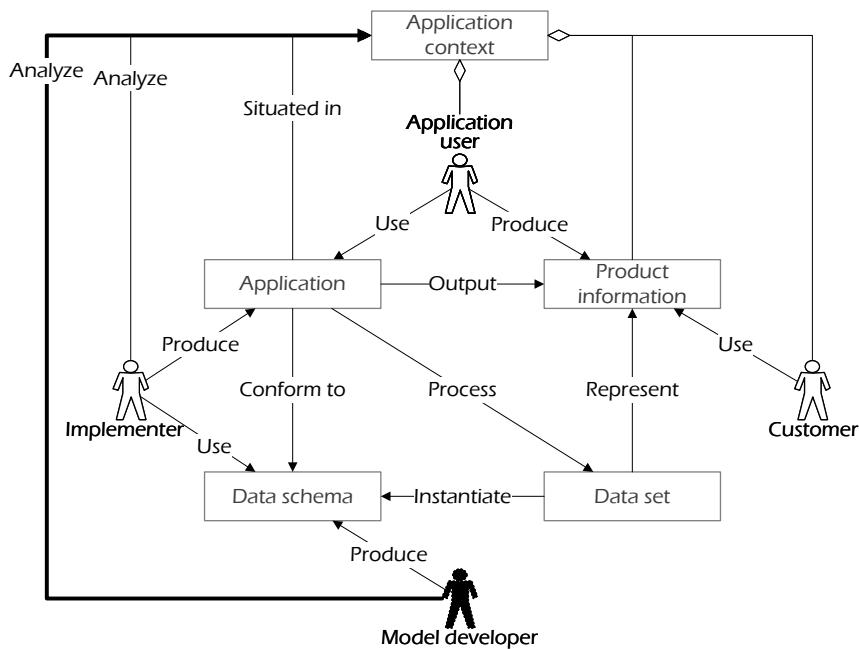


Figure 2.13 Application context, the current basis for both application development and information model development.

As stated in the beginning, methodologies to use application contexts in application development are mature in the domain of software engineering and HCI. It is time to look at how to use the application contexts in information model development, which has been described briefly in Section 1.2. A general development process is displayed in Figure 2.14. At first, activity models are used to describe application contexts. Developers elicit

information requirements from the activity models and create reference models to formally specify information requirements. In the end, according to the reference models, data schemas are developed to satisfy all the information requirements. This mapping process is a theoretical generalization of STEP modeling architecture which will be described in the following subsection.

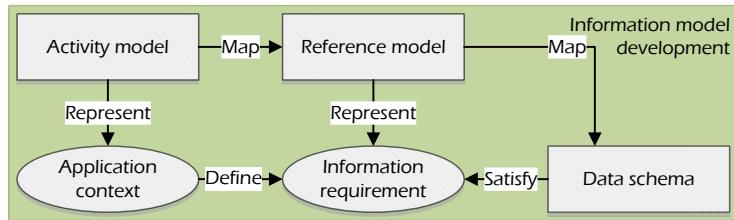


Figure 2.14 Application context used for information model development.

2.2.2 The STEP modeling architecture

To discuss behaviors and structures of information models, this study needs a reference widely-accepted in production engineering. For this purpose, this study uses a standard family, ISO 10303 STEP and related ones (e.g. ISO 13399, ISO 13584 PLib and ISO 14649 STEP-NC), as a main utility. Within the standards, almost all relevant semantics of product data have been standardized. Moreover, this standard family is known of a comprehensive modeling architecture formally specified and documented: It includes modeling languages, specification of application contexts, specification of information requirements, computer interpretable data schemas, development methods, implementation methods, validation methods, etc.

Modeling architectures are guidance of development and use of information models. For development, the traditional modeling architecture (ISO, 1994) is illustrated in Figure 2.15. It is a basis to derive other new modeling architectures (ISO, 2016c), e.g. an MIM (Module Interpreted Model)-based architecture constructing AP242. In a modeling architecture, the main goal is an exchangeable data set (in the bottom of Figure 2.15) conforming to an AP (Application Protocol). It instantiates a (part of) AIM (Application Interpreted Model) which integrates resource constructs to

satisfy information requirements represented by ARM (Application Reference Model). This mapping process is called “interpretation” which contextualizes resource constructs for particular information requirements. To represent information requirements, the ARMs use UoFs (Units of Functionality) or other modularization techniques to categorize conceptually interrelated application objects. In a well-documented AP, a set of AAMs (Application Activity Models), usually in a hierarchy formatted in IDEFo (Integrated computer aided manufacturing DEFinition for Function modeling), describes how the UoFs or the application objects work in an application context. Moreover, syntax, in terms of modeling languages, should be defined, e.g. EXPRESS (ISO, 2004) for ARM, AIM, interfaces and conformance testing; IDEFo for AAM; and p21 (ISO, 2016a) or p28 (ISO, 2007b) for data sets. In this way, a neat mapping process is defined from application contexts to data sets.

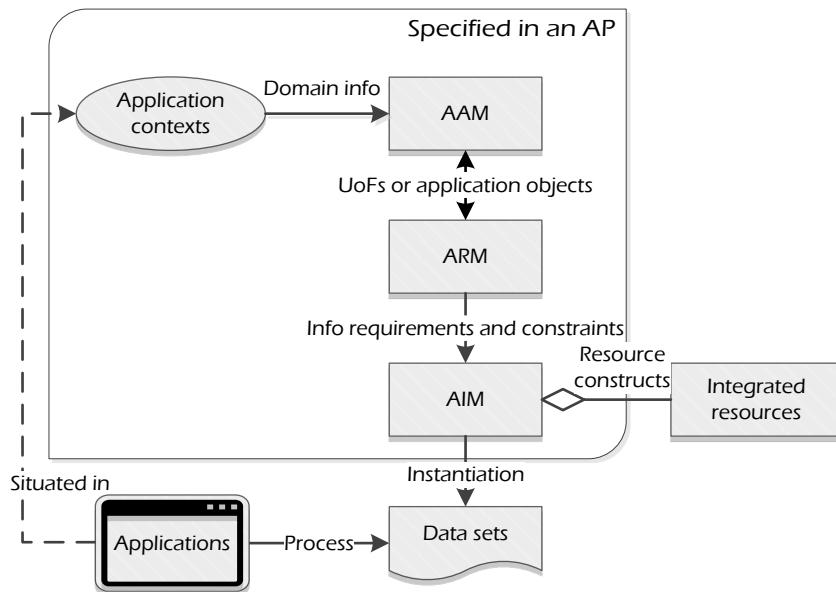


Figure 2.15 The traditional modeling architecture of ISO 10303 STEP.

This relatively complex structure of modeling architecture can be understood from a semiotic perspective, where semantics and syntax have been well defined. It is necessary in that engineering activities for product realization in a digital context relies on precise processing of tremendous amounts of information. In such a domain, it is often impossible for human beings to effectively manage and parse information entities (universals and particulars) without computer assistance (Arp, et al., 2015). Standardization of information models up to a semantic level is a critical strategy to facilitate consistent interpretation by computers. In a professional domain, standardization for interoperability at a semantic level or higher is difficult (Ouksel and Ahmed, 1999). The semantic quality of a model is “how well the model corresponds to the domain” and two important components are completeness (broadness plus granularity) and validity (Lindland, et al., 1994). High semantic interoperability demands a high level of formality (Ouksel and Sheth, 1999) to specify broadness, granularity, and validity. In STEP modeling architecture, AAM (for broadness), ARM (for granularity), and modeling languages (for validity) exhibit the formality.

Besides development of a data schema, the modeling architecture is also about use of a data schema by implementers. To help implementers, artifacts more than models are provided (Figure 2.16). In this illustration, general terms defined in Figure 2.14 are specialized in the STEP context. Conformance test tools and SDAI (Standard Data Access Interface, ISO, 1998a) are the additional for implementers to interact with data sets conforming to the data schemas (AIMs).

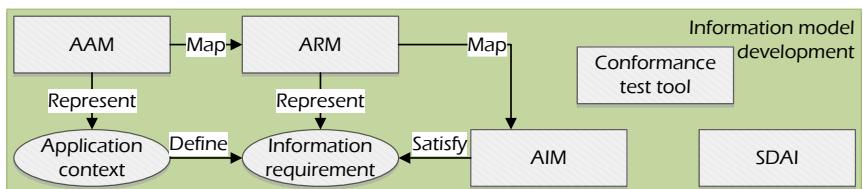


Figure 2.16 Artifacts (rectangles) in STEP modeling architecture useful for implementers.

Implementers' final goals are 1) to locate useful parts (the smaller block named “Useful part of AIM” in Figure 2.17) out of data schemas

(AIMs) and 2) to construct application software conforming to these parts of data schemas. For the first goal, implementers should find out a generally useful part of AAMs which can describe a context of use where the envisioned software is situated. Then implementers should identify a suitable set of ARMs, i.e. a set of information requirements corresponding to content requirements; this identification cannot be fully based on the selected part of AAMs because in most APs the mapping between AAM and ARM is not formalized. In the end, a valid part of data schemas (AIMs) can be obtained based on the set of ARMs. For the second goal, implementers should learn to use SDAI and conformance test tools combined with the part of AIMs, in order to construct the application software. The SDAI processes data sets at a syntactic level; namely, semantic correctness of processes cannot be ensured. Hence, conformance test tools are necessary to guarantee semantic correctness, i.e. completeness and validity. Issues regarding this implementation procedure will be discussed in Section 3.2.

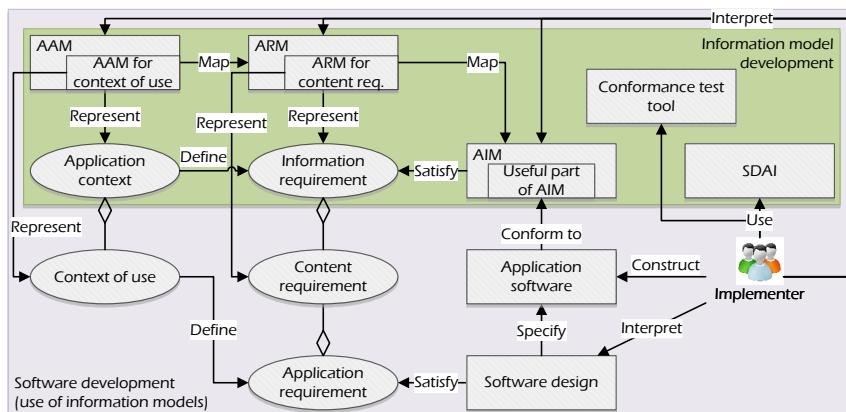


Figure 2.17 Software development based on the STEP modeling architecture.

One example of the above-mentioned procedure to generate a useful part of AIM is discussed in paper A where basic kinematic data exchange applications have been implemented. A beginning point is a CAD plugin (for Siemens NX) to export/import kinematic information; it is also required to be seamlessly integrated with geometric information in a form of

STEP AP214. In practice, the application occupies a small part of the information flows in the application context of AP214. All useful concepts for the kinematic modeling are collected in one AAM-data class and is used within one UoF (Kinematics, K1). This AAM-data class is referred to in two defined information flows: Concepts and product description, as outputs of two processes: Styling product and designing product, both of which are also introduced in ISO (2014a). In UoF K1, a list of application objects (ARM entities for unique concepts) is available, most of which form the needed segment of ARMs for content requirements of the application (e.g. kinematic joints, kinematic links and kinematic pairs). In the end, a useful part of AIM can be generated from associated mapping tables of these application objects.

Software construction (a late stage in software implementation) is introduced with a case in a part of Paper D, where use of SDAI and conformance test tools are discussed. SDAI is the most direct information models and the designated user interface for implementers. In the procedure of implementation (Figure 2.18), a useful part of AIM is actually an indirect resource used by implementers. The procedure begins with selection of APs potentially useful for the context of the developed software. A useful part of AIM is one result which should be manually obtained by implementers based on model artifacts (AAM, ARM, AIM, etc.) of APs.

Meanwhile, the AIM as a data schema can be compiled automatically by a SDAI toolkit (there is also a brief introduction of toolkits in paper D). Then, implementers can start programming supported by the compiled schemas and general operations provided by SDAI. A resultant program should be tested through data set validation against the compiled schemas.

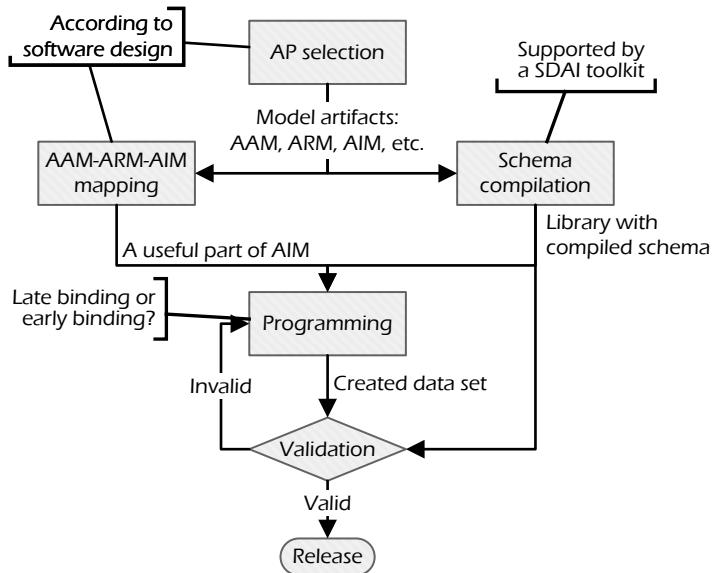


Figure 2.18 A workflow to implement information model standards based on STEP modeling architecture.

Note that the information models should not affect effectiveness of implementation. No matter how broad a standard is defined, it is still possible that its designated application context cannot cover all the implementation needs. Especially in an interdisciplinary scenario of system integration, information models are sometimes used in a context that are not pre-defined. It is also possible that different protocols are used together, so that a holistic view of information can be presented to application users. For instance, to enable interoperable classification (paper B) and categorization (paper C), engineering applications need collaborative operations from several sources, i.e. manufacturing resource (e.g. cutting tools) models to define semantics of dictionaries or catalogues, process models to specify usage of the resources and product design models to describe the products to be realized (the left side of Figure 2.19). These kinds of tasks require more than one type of information models to be integrated and implemented. Meanwhile, the right side of Figure 2.19 illustrates user needs for multiple

viewers to process information holistically from different viewpoints. Each viewer represents a style to present data the information models convey.

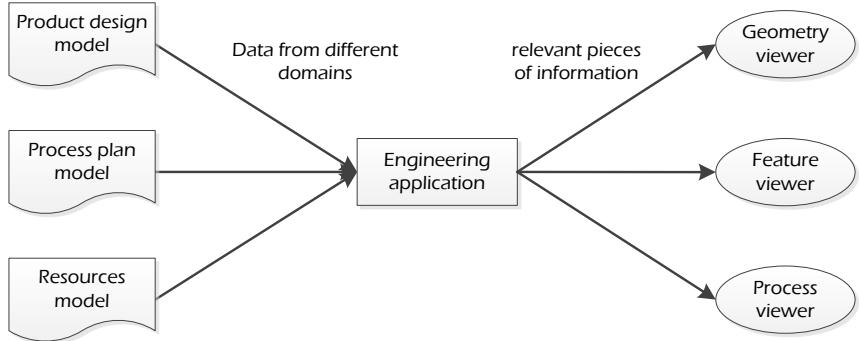


Figure 2.19 Information models from different domains are used by applications focusing on applying multiple views of product data.

2.2.3 Extensibility and portability of STEP

This modeling architecture demonstrates the four major requirements of an information model: Extensibility, portability, interoperability and longevity (Al-Timimi and Mackrell, 1996). For the scenario of system integration, interoperability is why a standardized generic information model is needed and selected in the first place. Longevity of an information model depends on how accessible and reusable it is, which is all about the interoperability and portability. Thus, extensibility and portability become two most critical qualities that determine how valuable an information model is for users and how interoperable and long-lived it is eventually.

Al-Timimi and Mackrell (1996) defined the *extensibility* as an ability to “continue to take advantage of new and innovative techniques” emerging from continuous evolution and the *portability* as an ability to “move data among applications”. More generally, ISO/IEC/IEEE (2010) defined extensibility (i.e. “extendability”) as “the ease with which a system or component can be modified to increase its storage or functional capacity,” and portability as “the ease with which a system or component can be transferred from one hardware or software environment to another”. It is clear that extensibility is about internal structure of a system and that portability

is about external use of a system. Both qualities are useful for system integration, in that extensible information models can easily integrate information from different contexts and that portable information models can be easily reused in different applications.

In other words, a changing environment is the root cause of “extending” or “porting” an information model. An information model can take effect only when it works in a using environment. Application contexts and applications are two major constituent elements in this environment. It is the various application contexts and the various applications that constitute the changing environment. Any information model has a potential to be used among many other applications in a designated (or overlapped) application context (Xu, 2009). However, again, it is vague to understand the extensibility and the portability without distinction of data schemas and data sets when it comes to applications and application contexts (Figure 2.20).

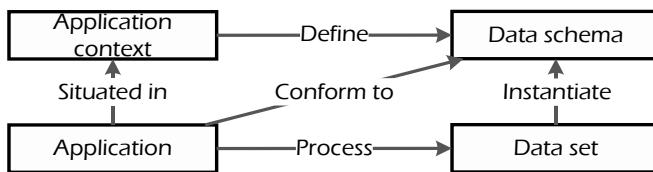


Figure 2.20 A data schema and a data set.

An extensible information model should facilitate efficient data schema extension to meet new information requirements. The new requirements for system integration usually cause changes in processes and information flows, i.e. the application contexts and thus require an extension of data schemas (Figure 2.21). It is also possible to extend data sets, but how much a data set can be extended is delimited by the instantiated data schema, so this thesis only concerns extensibility as a preferred quality for data schemas. Besides, backward compatibility is also an issue for extending a data schema, i.e. the extended information model will still be effective with the applications complying with the previous version of the information model.

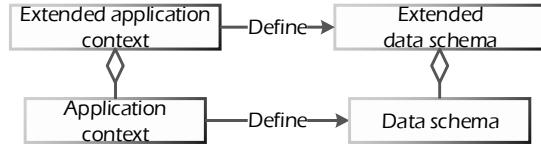


Figure 2.21 Extending the data schema based on information requirements from new application context.

On the other hand, a portable information model should facilitate efficient reuse of information represented in a data set by a new application. The data set may be created or exported in another application complying with an instantiated data schema. Although porting the data set is most important for application users, implementers should also, in advance, port the instantiated data schema to the application to automate the reuse of the data set. It is the portability of an information model that determines the ease of making the new application conform to a data schema and of interpreting a data set (Figure 2.22). A simply porting process reuses an existing data set without any change in content and the porting process is executed during an implementation process. The extending process and the porting process can be easily distinguished by whether there is a change in the content of the information models.

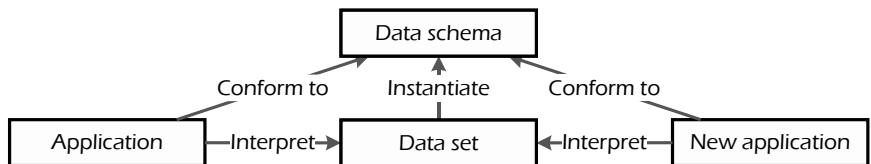


Figure 2.22 Porting the information model to new application.

Hence, these two abilities actually take care of different stages of the technical process of the information models, i.e. the development stage and the operation stage of an information model. The data schema and the data set are two results for the corresponding stages, respectively. Extension of content of the data schema is the major process to be evaluated for extensibility. Evaluating portability should evaluate the process performance of porting the data set and the instantiated data schema in different applica-

tions. There are different aspects to describe the performance when evaluating the abilities. Efficiency and effectiveness are important metrics to evaluate the extent of the two qualities. Efficiency is obviously a major focus to evaluate extensibility and portability, if high delivery accuracy of the extending process or the porting process is assumed.

2.2.4 High extensibility is preferable

High extensibility is a preferable quality for information models for many reasons. Model developers decide what quality an information model should possess. High extensibility is obviously prioritized since it is highly related with the ease to develop information models. Besides, information models are expected to support an application context as broad as possible to enable high readiness for system integration. An extensible model is hopefully easy to be reconfigured to meet new requirements in an extended application context.

There are plenty of implemented approaches to enable high extensibility in general. Agile modeling strategies to enable development of information models to meet changing requirements efficiently can be traced back to standardization of database management systems, with three schemas for presentation, logic and representation (Tsichritzis and Klug, 1978). Therein models can evolve independently of applications. The idea of decoupling contexts from data is an important step to enhance extensibility. According to this decoupling principle, an information model should be independent of the applications using it. In other words, what the model is used for is more of concern than how the model is used. Besides, an information model should be constructed inherently with reusable building blocks. The reusable building blocks are information representations at different levels of abstraction. STEP AP242 constructed by MIMs (Model Interpreted Model) is one example with a hierarchical set of building blocks. With this manner, different application contexts can be supported with different composing strategies (Figure 2.23). One building block may be constructed with other building blocks at a higher level of abstraction and a lower level of hierarchy. The dashed line in Figure 2.23 indicates that a data schema describes a part of an application context, in terms of information requirements. Last but not least, an application context defining an information model should be broad, coherent and semantically closed so

that possible instances (the data sets) have sufficient extensibility and that the interoperability of the using applications is not constrained by the data schema too much.

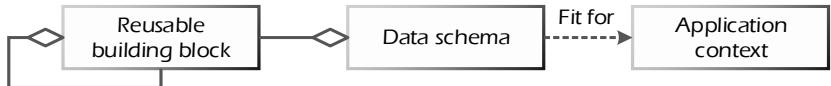


Figure 2.23 Reusable building blocks to compose an information model.

Just like any other types of systems, portability and extensibility are non-functional requirements stated informally and sometimes controversially (Mylopoulos, et al., 1999). The portability requires an information model to care about its accessibility and reusability by a specific application (with data sets) or a specific implementation practice (with data schemas). In other words, how to use and reuse the information models matters for portability. High portability requires efficient implementation of a data schema in an application so that valid data sets can be produced or interpreted by the application. This process is usually performed manually by implementers. Hence, the modeling architecture (probably in a complex hierarchy with building blocks at different levels of abstraction) should be either easy to comprehend by implementers, or hidden with a simplified implementation architecture. Moreover, an application is not necessarily designed to meet an application context as broad as supported by a data schema. Implementers are required to specialize and contextualize a generic data schema in accordance with implementation technologies and specific application requirements. These context-dependent requirements by high portability is difficult to be met or easy to be ignored by information models that typically have high extensibility. In general, requirements derived based on extensibility and portability are often unrelated, distinct and even contradictory, as described in Table 2.1.

	Extensibility	Portability
Scope	A broad application context	A specific application or a specific implementation
Concern	What the model is used for	How the model is (re)used
Structure	Composable structure	Comprehensible structure
Behavior	Design for extending	Design for porting (reuse)
Stage	Development	Operation

Table 2.1 Different requirements on information models by extensibility and portability.

2.3 A model driven approach

The model driven approach focuses on integrating product data from different sources and reusing it for different application contexts, during the course of product realization. A system based on this approach should keep information in context, versionable, associative and retrievable in information models (Nyqvist, 2008). A model driven system interprets and processes data directly from information models rather than unstructured documents that are difficult to integrate and reuse. This mechanism relies on information models as a basis not only to represent and exchange information, but also to facilitate coherent interpretation and processing. The model driven approach is a typical composable solution that does not rely on context to an extent.

2.3.1 Contextual proprietary information models

It can be observed that product data, as valuable intellectual resources in the engineering activities, is constrained by system that create it, as described in a tool-driven manner (Section 2.1.1). System vendors, instead of system users as data owners, control the ways to interpret, process and communicate the data. Accordingly, product data is modeled based on how the system is implemented, used and maintained. Schemas are directly based on internal data structure of the applications and thereby highly contextual. The produced data sets are typically proprietary lock-in, i.e. accessible only with specific applications or APIs provided by vendors. Although there might be APIs, a large number of details are hiding from users, which

makes it difficult to be reused by other systems. When the APIs are involved, it is useful to mention two types of programmability abstractions (Ford, 2013): Contextual and composable, with pros and cons to each. The common interfaces to current CAx systems are featured by the contextual abstraction, with the following advantages:

- High findability and feasibility for a specific problem.
- Concrete overarching templates.
- Being easy to learn and use.
- Being able to achieve specified user goals efficiently.

The benefits of the contextual abstraction apply well to information models. A contextual information model builds contexts into components, where it can provide “out of the box” data structure and methods for specific contexts that users search for initially. Thus, the contextual ones are easy to learn and use initially.

Contextual abstraction and composable abstraction

Ford (2013) proposed the two types of abstraction without formal definition. Abstraction can be seen as a selective technique to decide what should be visible and what should be invisible. There are different selective criteria to make the decisions. The criterion to distinguish contextual abstraction and composable abstraction is the implementation aspects. For an information model as an abstraction of product data, engineering activities using the data, i.e. application tasks, are an important concern. Another concern is about implementation activities. Different levels of abstraction according to existence of a concern about implementation activities can make a huge difference. *Contextual abstraction* takes the implementation context as a whole into account, which makes a large number of details in an information model related with implementation aspects. *Composable abstraction* does not consider the implementation aspects, which makes an information model contain few details about implementation aspects (Figure 2.24). In other words, the composable abstraction is independent of implementation details, so that it can be used to compose any implementation context with specific implementation aspects, which delivers high composability, but may require effort to make the composing happen.

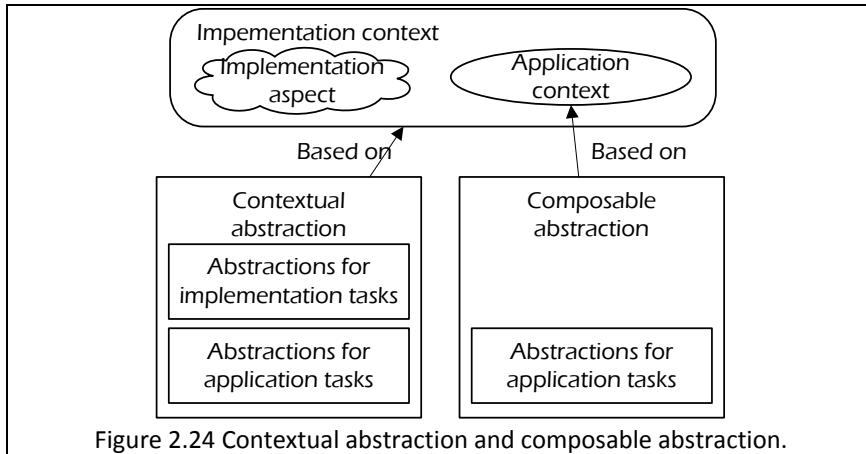


Figure 2.24 Contextual abstraction and composable abstraction.

However, when the information models are actually attached deeply with the systems, it is difficult to extend to other non-envisioned contexts, e.g. in other systems for similar problems. Besides, even though the solution is easy to learn and use, the users cannot reuse their experiences in other systems. Another major critique on the contextual approaches is the Dietzler's Law (80% of what users want is fast and easy to achieve, 10% is difficult and expensive and the last 10% is impossible, but the users demand 100%) when observing a project built upon Microsoft Access (Ford, 2008). The engineers required perfect artifacts to represent and convey their ideas, but the systems could only provide a nearly comprehensive solution through predefined processes and templates.

2.3.2 Composable standardized generic information models

On the other hand, standardized generic information models, as a basic enabler for the model driven approach, are a composable solution. They are decoupled from the contexts of specific systems. A standardized generic information model is not supposed to anticipate implementation details. The independency facilitates a high readiness for various contexts. It makes provision for possible use by different applications. Thus, a standardized generic information model can theoretically form a base to capture, integrate and represent information from different domains in different

contexts. For instance, the ISO standard STEP AP242 (ISO, 2014b) provides system neutral data structure that can be processed independent of applications and be reused by many different systems within a wide scope. When completely based on AP242, applications only provide functional support to accomplish engineering goals, rather than become technical limits to hinder communication.

Note that this definition of categorization for abstractions is relative. Compared with proprietary information models, ISO 10303 is more composable and less contextual, but specific definitions of syntax and semantics make it still contextual to some extent. The use of the “application context” suggests the semantic context. The syntax is usually formatted as an English-speaking-based “p21” file encoded in the ASCII standard. Hence, there are still some contextual prerequisites for a composable solution like STEP.

2.3.3 Combination of contextual and composable solutions

That said, the advantages of the contextual abstraction are compelling, which brings forward an idea of a new modeling architecture (Hypothesis I and section 3.5). The new modeling architecture adds a contextual layer to enhance the standardized generic information models without undermining the original composability. The contextual layer represents information requirements for specific implementation methods. It makes information models fast to learn and use by a certain user group for a certain type of implementation tasks. Still the composable part should be kept to address unanticipated use cases. Users (i.e. the implementers) should be allowed to use any composable components together with the contextual layer.

The combination of the composable and the contextual solutions explains the strategy of the model driven system integration developed in this study. The existing composable standardized generic information models drive how applications can function effectively, the contextual layer of the modeling architecture drives how implementations can be performed efficiently. Different stakeholders can benefit from the model driven system integration in several aspects:

- Interoperability: Standardized generic information models are used to represent domain information used in applications and is supposed

to be decoupled from implementation practices. Hence, contexts of application are far more emphasized than contexts of implementation. As a result, standardized generic information models can be contextualized for different engineering applications without losing consistency of representation. That is, implementation strategies do not hinder interoperability of data sets.

- **Independency:** Models act as a medium for application users to interpret and process information as their intellectual assets. Any specific vendor or platform is not supposed to limit use of the models. Engineers should be allowed to choose when, where and how to manipulate information. Standardized generic information models facilitate a basis for envisioned applications to be integrated within an existing ecosystem. However, independency is also a major obstacle of development of standard communities, because it does not benefit major system vendors who have significant influences on the industry.
- **Productivity:** A major issue regarding product information in industry today is that information is processed, stored and communicated in a fragmented manner, which results in the tool-driven situation. Engineers' workloads increase exponentially along with growing numbers of data sources, let alone inevitable information loss or errors during these efforts due to human errors and systematic incompatibility. Brunnermeier and Martin (2002) estimated imperfect interoperability costing about \$ 1 billion annually and at least two-month delay for new model introduction, in the US automotive supply chain. Gallaher et al. (2004) quantified annual inadequate interoperability costs in US capital facilities industry as \$15.8 billion. Therefore, as a basis of system integration, standardized generic information models have great potential to simplify the engineering activities and to increase productivity.

2.4 System architectures

It is a complex context that requires systems to be architected before detailed design and construction. Implementing system integration is naturally an interdisciplinary task, especially to satisfy complex but critical

business needs and technical needs in the engineering context. An architecture should directly address concerns of those stakeholders with distinct expertise who are inevitably involved in an entire lifecycle of an integration application. The stakeholders are a central part of the complex context, who possess knowledge, perform tasks and interact with systems. There are immediate stakeholders, e.g. designers, programmers, testers and users, who are most concerned about technical tasks, in terms of design, development, evolution and maintenance. There are also non-technical stakeholders such as customers and organizational managers who are indirectly influenced by system performance to accomplish high-level business needs. All the stakeholders are living in a constantly changing environment where technologies and business change frequently. The coordination of technologies, business needs and concerned stakeholders imposes great requirements on architectural design to deal with the changing circumstances.

An architecture defines overall system structure with a set of design decisions (CMU, 2016) which specify selection, organization and composition of architectural elements (Booch, et al., 2005), in terms of forms and rationales. An architecture mainly addresses concerns about functional requirements and quality requirements of systems. Quality requirements drive development of system architecture, especially for software-intensive systems (O'Brien, et al., 2007). The previous sections of this chapter have revealed a quality requirement mostly relevant for system integration, i.e. interoperability. An architecture targeted on the interoperability is critical for overall performance of system integration.

Architecture

An architecture, as a term used in many domains, describes underlying structure of a system (Brand, S., 1995). Architects propose solutions with balanced integration of durability, utility and beauty (Vitruvius, 1914). The ultimate goal of architecture is to manage complexity at an understandable level for concerned stakeholders. It is usually not necessary to draw 3D design, to plan a budget and to set up a schedule to build a simple doghouse (See the figure below). However, it is impossible to accomplish a complicated building without architecting to orchestrate an enormous number of factors and correspondences. An architecture helps people to reuse previous knowledge of methodology to solve problems at

hand, to reduce complexity with reusable design principles and to fulfill certain requirements (Spinellis and Gousios, 2009).



A doghouse
(<https://pixabay.com/en/photos/cartoon/>).



Empire state building
(<https://pixabay.com/en/photos/empire-state-building/>).

An architecture of an information system also shares a similar definition. ISO/IEC/IEEE (2011b) defined an architecture as “fundamental concepts or properties of a system in its environment embodied in its elements, relationship, and in the principles of its design and evolution.” It can represent a process from requirements to a final developed system (Bass, et al., 2012). Description of an architecture shall be a core work product to enable communication and cooperation among stakeholders (ISO/IEC/IEEE, 2011b).

Several basic techniques are important for successful architectures. Modularization with prudent abstraction forms a major means to reduce system complexity and increase architectural agility. Stakeholders need modules to understand, develop, use, or maintain the systems. All the major architectural patterns, e.g. layered, event-driven, service-oriented and pipelines, are about modularization. The most fundamental principle for architecting is high cohesion within a module and low coupling between modules, which leads to concerns of other internal quality attributes for implementation, i.e. reusability, readability, maintainability and changeability. Examples of modularization of an architecture are a layered pattern

describing a modeling architecture in Section 3.5 and an MVC (Model-Viewer-Controller) pattern to specify an integration architecture in Section 4.9.

Architecture pattern and architecture style

Changes happen all the time during an entire system lifecycle. Decisions on a system architecture are all about compromise. That is, a change in one feature often impedes another feature (Spinellis and Gousios 2009). It is a good design of a software architecture that guarantees changes to cause minimal impediments. Patterns provide general answers on how to reach good designs. Alexander (1979) defined a pattern as a three-part statement: “A certain context, a problem, and a solution.” It is the patterns that present well-proven schemes guiding architecture design for recurring problems in certain contexts (Buschmann et al., 1996).

A style is a descriptively and prescriptively codification of a set of architectural design solutions with certain commonality (Perry and Wolf, 1992). Unlike patterns, such codification is often incomplete for any specific problems (Bachmann, et al., 2011). Similar to styles of building architecture, software architecture styles are more useful for categorization and documentation, rather than problem-solving.

In particular, the MVC (Model-View-Controller) pattern is chosen for its specialized suitability for interactive systems with heavy needs of coordinating data representation and presentation. The MVC pattern is a decent way to achieve loose coupling between data models and UIs (User Interfaces). It highlights reusing data for different contexts as a key feature of any interactive system. An information model is usually easy to be presented where it is created, but can be useful only when reused, integrated and presented in other contexts. MVC is a basic infrastructure to facilitate reusing of data. This thesis will present how to apply the MVC pattern to implement the information models as a basis to integrate and reuse engineering data in Section 4.9.

An implementation process delivering stage-by-stage architectural decisions is also challenged in this particular context. Formal information models are often specified with syntax uninterpretable by human beings, which affects implementation for readable presentation. Fortunately, the

interpretation of the information models is possible for computers. Thus, there should be a specific implementation guidance to translate the computer-interpretable representation to human-interpretable presentation. Common development solutions like agile development methodology with iteration based on prototypes or increments is often criticized for insufficient predictability and high expense, and, on the other hand, Normal waterfall-style sequential development process is too rigid to adapt changing requirements and inadequate design in reality. It is promising to compensate these methodologies with contextual interpretation of models, as proposed by the Hypothesis II, to reach a system architecture design with relatively low complexity and high implementability.

2.5 Information architectures

In an information model, a large amount of information is usually integrated according to a specific scope and related requirements regardless of usefulness for particular processes or applications. It has to be done this way to keep completeness and consistency of the data and thereby to ensure high readiness for different applications. The mechanism is guaranteed by regulated syntax, description methods, interpretation methods, implementation methods and validation methods. The resultants of completeness and consistency contribute to high semantic interoperability and high computer interpretability. Naturally, human interpretability of information models is ignored by the complicated mechanism. Let along data amount is often too large to be cognitively processed.

The major objectives of digital technology are to augment human intellectual abilities and to compensate human limitations in practice (Bush, 1945), rather than to stand in the way. In the engineering context, the augmentation is greatly influenced by data generated through a full product life cycle, which can be used for many purposes: To automate processes, to simplify activities, to convey knowledge, to evaluate ideas and to make decisions. Information models make the data interpretable by computers, but it is the responsibility of digital applications to make the data applied effectively for the contextual purposes, i.e. to augment human intellectual abilities. In fact, similar stories have happened everywhere in contexts of integrating and sharing information. Data should not just be organized to make applications work well, but also to facilitate ease of use by human

users. Thus, a good interaction design to present information, guided by an information architecture, should be delivered to let information make sense to users.

ISO/IEC (2010) defined *information architecture* (IA) as “(human-centred) structure of an information space and the semantics for accessing required task objects, system objects and other information.” Practicing IA is all about organization and description of content in terms of objects or items in a principled space. Patterns and sequences are two most important aspects to organize and describe content (Garrett, 2010). There are two general objectives of designing IA: 1) Help users accomplish their goals by easily navigating content and locating objects and 2) Help vendors accomplish business goals by educating, persuading and notifying users.

Today, UX (User Experience) designers for everyday commercial applications usually consider IA as a part of user experiences, which is regular but not dominant. After all, it is not worthy to put much effort on architecting information that is not so sophisticated, when attracting and engaging users is more prioritized. However, most engineering applications are characterized by overloaded information, burdensome cognitive tasks and demanding domain expertise (Redish, 2007). These complex professional systems usually have a small user group and are rarely encountered by the HCI (Human-Computer Interaction) experts. Research on complex systems obviously lacks economy of scale and, therefore, the HCI community is less likely to consider such scenarios.

Engineering CAx systems are typical content-heavy applications. Information from different sources in different forms are required to be integrated and to be used for many kinds of processes. As implied by the naming, at most time CAx systems aid engineers to accomplish some goals, rather than to perform the jobs automatically. The aid starts with visualization of overloaded information and ends with a conveyer to communicate information with other systems or other human users. The troublesome tasks with high requirements on efficiency and effectiveness make engineers lack time to learn patterns of interaction and presentation as they might succeed in normal applications. Hence, architecting information can provide a way for users to easily interact with information with organized and clarified content and processes (Spencer, 2010).

The overall system integration is facilitated at different levels. Applying information models is not only connected with specific systems, but also connected with specific business logics and workflows in an interactive environment. At a business level, a system integration solution is heavily determined by qualities of interaction, e.g. findability, usability and learnability. Standardized representation enables interoperability at the levels of syntax and semantics. In system integration, properly architecting information improves interoperability to a structural level. At this level, the information representations are presented with patterns and sequences to ease the human interpretation.

Chapter 3

A new modeling architecture

*All problems in computer science can be solved
by another level of indirection.*

- David J. Wheeler

This study aims at a practical solution for efficient system integration to overcome the tool-driven manner (Section 2.1.1) and to effectively support product realization. The solution should support conventional implementation practices and fix the efficiency issue caused by the focus on application contexts (Section 3.2) when developing information models. It requires a new modeling architecture (Section 3.5) to facilitate high pragmatic interoperability (Section 3.1). Implementation contexts (Section 3.3) and implementation models (Section 3.4) are new concepts introduced in this study as imperative parts of the new modeling architecture.

3.1 Envisioned pragmatic interoperability

Anything for communication has its semiotic aspects. In the Information Age, syntax may remain as a stable concept, but semantics and pragmatics have been evolving with the employment of computers (see Section 2.1.3). Within this semiotic framework, computer interpretation requires a decent semantic level. As introduced earlier (Section 2.2) semantics of information models has been defined very well. Ideally, model developers create semantically perfect data schemas so that all conformant data sets can be automatically processed and effectively reasoned upon by computers.

Nevertheless, human interpretation of the information models is still inevitable, especially for implementation, which requires pragmatic interoperability of the information models as well. Griffin, et al., (2002) proposed the highest level of data interoperability as seamless sharing of information, in a form of common exchange structure as well as universal

interpretation. The human use of information models should not only focus on representation for effectively capturing and integrating information from different sources, but also on ease of human use. However, existing interfacing techniques do not support human interpretation sufficiently. Thereby, enhancing the pragmatic interoperability is the focus on the new modeling architecture.

This study claims that the implementers are the core stakeholders for human interpretation of information models. It is critical to emphasize the human audience when studying the pragmatics of any kind of representation. Superficially, pragmatic interoperability of an information model should be facilitated for scenarios of both development (with data schemas) and use (with data sets) of applications. In other words, implementers can easily use data schemas and application users can easily use data sets. Figure 3.1 illustrates the use of the two types of information models (data schemas and data sets). Figure 3.1 also shows that an application is an unavoidable media for use of a data set and the application users actually do not interpret any form of information models directly. Applications, developed by implementers, control all pragmatics for application users. Therefore, in fact, the implementers perform the only direct human-model interaction.

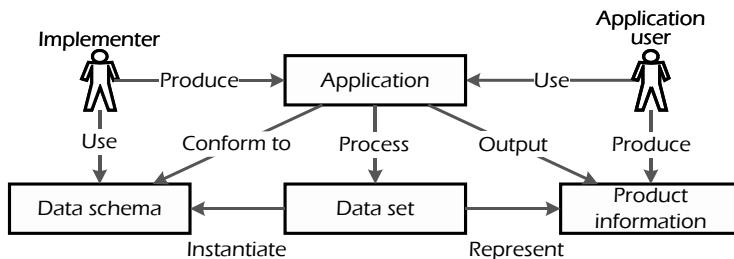


Figure 3.1 Pragmatic interoperability that should be addressed by information models.

The implementers' role hereby makes implementability of an information model, for either system integration or new functional development, exhibit pragmatic interoperability. Taking kinematics as an example, kinematic modeling (paper E) and kinematic error modeling (paper A) have been studied and implemented based on STEP. Within the scope of

STEP, the first edition of a data schema for kinematics was published as a resource with decent completeness and validity in 1996 (ISO, 1996); AP214 (STEP Part 214: Application protocol: Core data for automotive mechanical design processes, ISO, 2010a) was the only application protocol integrating it; the first published data set was done by Hedlind, et al. (2010) and the first published conformant application was done by Li, et al. (2011); no commercialized application has been developed until the second edition of the data schema was published (ISO, 2014a). It might be difficult for average implementers to use the data schemas as what Hedlind, et al. (2010) or Li, et al. (2011) did. Same things happen when implementing new functions (rather than just integration): No one would expect a CAx solution for kinematic error analysis, without high implementation readiness of related information models (paper A); no one would expect digitalization of product catalogue communication, without high implementation readiness of standardized digital catalogues (paper C).

Not only implementability related, portability and usability of information models are also in relation to pragmatic interoperability. As introduced in Section 2.2.3, portability is an important quality yet not very well satisfied and high pragmatic interoperability of an information model can lead to high portability. With a modeling architecture to facilitate human interpretable information models, it is easier for implementers to use data schemas in new contexts, which means easier to “port” the data schemas in new contexts. In turn, high portability exhibits high pragmatic interoperability. Moreover, in its own right, portability of an information model is equivalent to ease of implementation (implementability). From the perspective of implementers as users of an information models, portability is also equivalent to usability in this context. Thereby, with implementers as direct users, pragmatic interoperability of an information model facilitates its portability, usability and implementability (Figure 3.2).

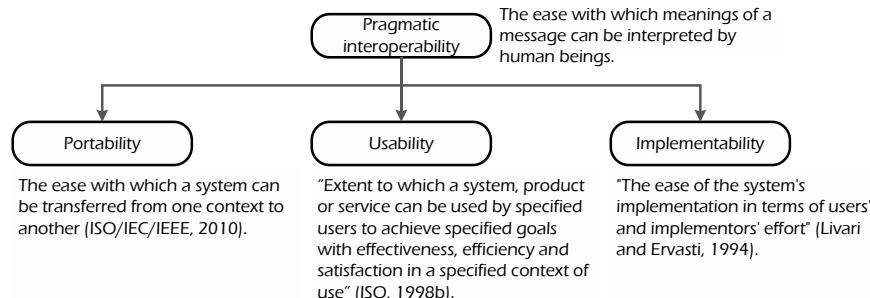


Figure 3.2 Pragmatic interoperability of an information model facilitates portability, usability and implementability with implementers as direct users.

3.2 Application contexts: A pitfall

Any architecture shall include a correct definition of a context as an initial part, so shall a modeling architecture. From a semiotic perspective, a context is where interpretation of an artifact happens. Understanding of the context helps developers know what kinds of artifacts are best for interpretation in the specific environments.

3.2.1 Application contexts ignoring implementers

Section 2.2.1 has introduced application contexts as a current initial point for both software development and information model development. The application contexts help developers in both domains know the interpretation needs of application users. Particularly for information model development, understanding of the application contexts has been beneficial for standardization of semantics (more details in Section 2.2.2).

Nevertheless, is an application context sufficient for initialization of information model development (Figure 3.3)? The answer is no, in that the implementers, instead of application users, become the direct and primary user group of the final products (data schemas). The final products are not used in the application contexts. Hence, the application contexts fail to represent the environment where the final products are used. In a word, using application contexts as the initial part of a modeling architecture violates the reason to specify contexts for development of an interactive system.

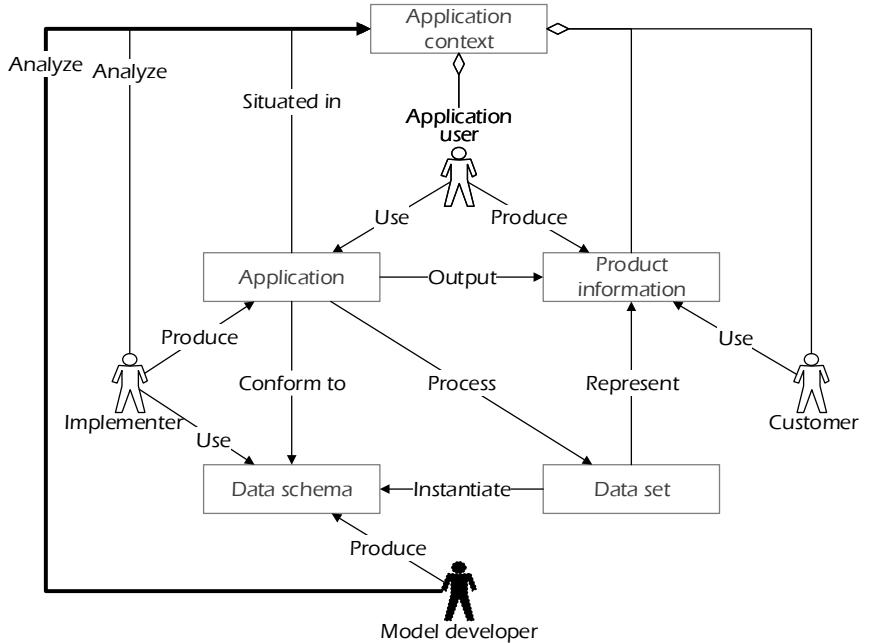


Figure 3.3 The current situation: The application context forms a basis for application implementation as well as model development.

3.2.2 A different implementation workflow

This misplaced focus on the application contexts leads to difficulties faced by implementers. One consequence is that involvement of information models makes an implementation workflow significantly unconventional.

How to implement an information model has been introduced in Section 2.2.2. Taking ISO 10303 STEP as an example (Figure 3.4), model developers should generalize all possible use cases in application contexts and map into a comprehensive list of information requirements. This mapping in model development is performed between the AAM and formalized ARM, in order to create a possibly very huge AIM for a generic information standard.

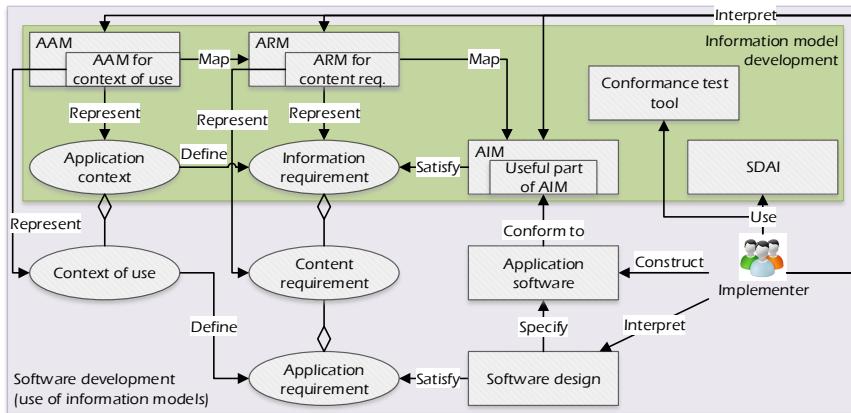


Figure 3.4 The traditional way to implement an application conforming to a STEP data schema.

Hence, for implementers, locating a valid segment of a data schema (e.g. the AIM) is an unavoidable task. The task needs AAM and ARM as intermediates to manually interpret AIM (arrows from the implementer in Figure 3.4). Moreover, the model artifacts (AAM and ARM) convey information (e.g. contexts and requirements) that may be familiar for implementers, but the syntax is unfamiliar. The difficulty is undoubtable to consistently map one unfamiliar representation to another (Gielingh, 2008).

This mapping mechanism makes the implementation process greatly entangled with the information model development process (illustrated by the two biggest blocks in Figure 3.4). These practices are neither intuitive nor efficient for implementers who are direct users of model artifacts in a modeling architecture. By manually interpreting most model artifacts, implementers, as users of a system (i.e. the AIM), practically learn how this system is developed. This manner is not reasonable for design of any kind of interactive systems. Just like few software users can understand how software is designed and developed, few implementers can do the same for information models.

3.2.3 Increasing complexity of modeling architectures

Besides, each generation of the STEP-based modeling architectures tends to increase the number of model artifacts (Figure 3.5). The evolution in Figure 3.5 suggests that more and more layers and modules are involved for each modeling architecture. It is caused by a critical need to support rapid model development in the changeable application contexts (Section 2.2.4). Hence, model developers need a flexible structure to facilitate rapid standardization and backward compatibility, i.e. 1) extensible definitions of the application contexts, 2) reusable semantic constructs and 3) standardized syntax with designated access interface. Eventually, the use of data schemas (e.g. the AIMs in Figure 3.5) are instructed with complex hierarchical structures.

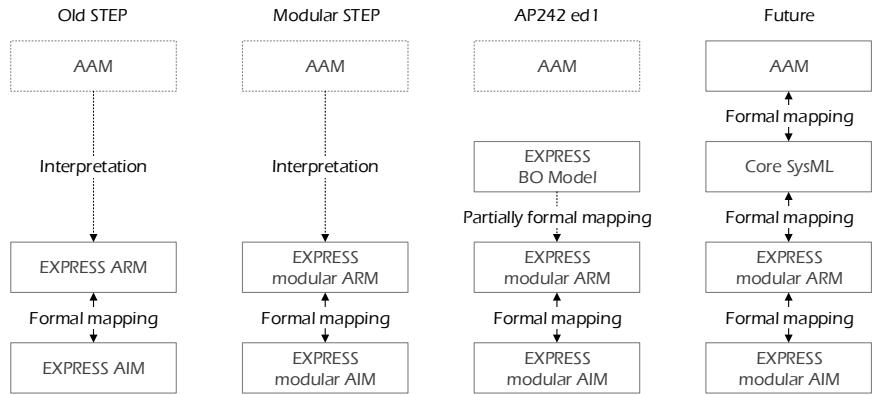


Figure 3.5 Evolution of the modeling architectures, based on ISO (2016c).

It results in more and more complex structures applied in implementation projects. These structures are problematic for the implementers as direct users and interpreters. Implementers need to learn different mapping mechanisms when different data schemas are applied. They need to interpret piles of documents to specify information models at different levels of abstraction. The appended paper D describes a case study for this issue during implementation.

3.3 Implementation contexts: The new concern

Therefore, an intended context where an information model is used should be sufficiently defined in the modeling architecture which specifies development of the information model. In general, a context should describe “relationships, dependencies, and interactions between a system and its environment” (Rozanski and Woods, 2012). For an interactive system, the context should mainly describe the interaction between primary users and the interactive system. This is why an application context can form a basis to develop applications. However, for an information model, the primary user is who use the data schema directly, i.e. the implementers, in that any use of information models begins with implementation.

3.3.1 Defining implementation contexts

With this knowledge, the context is where implementers use an information model to implement an application, which is named as an *implementation context*. Note that the information model in this definition means standardized generic information models interacting with implementers with model artifacts, e.g. data schemas, reference models and data interfaces; examples are widely accepted Application Protocols (APs) of ISO 10303. Like application contexts, an implementation context is a part of a modeling architecture where it guides development of information models by describing the interaction between implementers and information models (Figure 3.6).

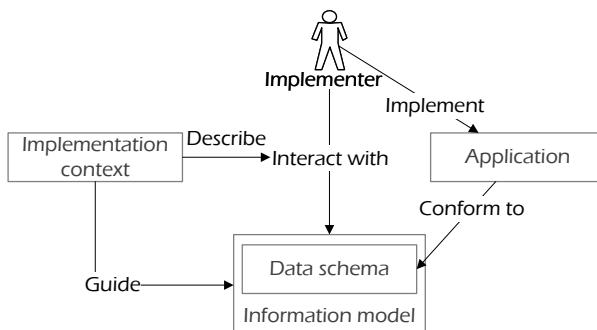


Figure 3.6 The usage of the implementation context.

For a modeling architecture, an implementation context includes implementers and applications (the implementers' primary goal) as major components (Figure 3.7). Identification of implementation contexts addresses the implementers as the primary users of information models, in development of modeling architectures. This identification of an appropriate audience is a basis of any discussion about pragmatics and the context of the pragmatics. Any developer shall clarify this identification to know whether their creation is useful and usable. As a result, a modeling architecture guided by an implementation context can deliver data schemas appropriate for implementers.

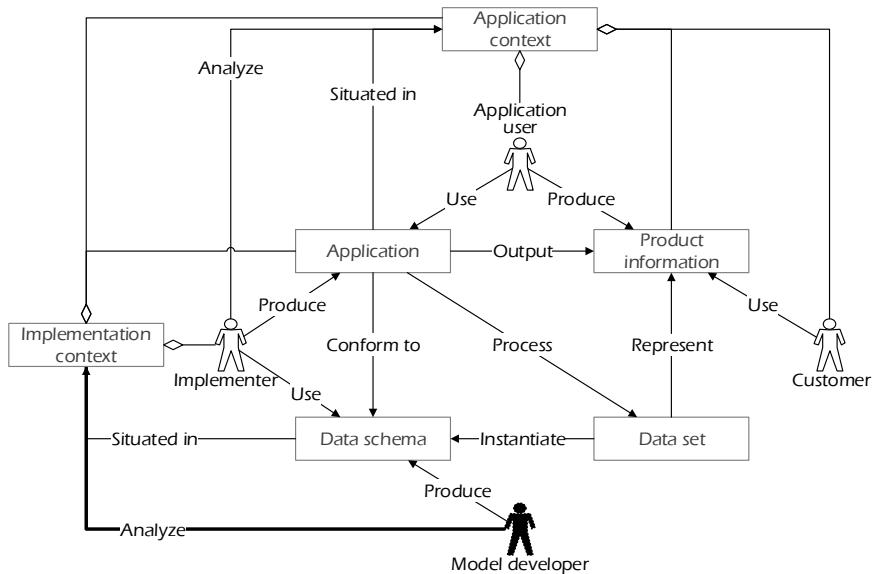


Figure 3.7 Implementation context, a shifted basis where information model development begins.

As also illustrated in Figure 3.7, the originally-defined application context is a part of the implementation context. This is a result of taking into account the entire software implementation process where implementers aim at problem solving in the application context. Including application contexts to a certain detailing level is how implementation contexts are

concerned with indirect elements. Comparably, a typical application context also includes indirect elements (e.g. needs, characteristics, tasks and knowledge of the customers).

The inclusion of application contexts suggests that boosting portability of an information model shall not jeopardize its extensibility (more details in Section 2.2.3). It also suggests a layered structure where one layer encapsulates another for different purposes. The original modeling architectures are kept to maintain extensibility, but a new layer should be added to facilitate portability. This new layer is what will be introduced in Section 3.4.

To describe the implementation context in detail, the following subsections will make a categorized discussion about the relevant contextual components based on paper D. The categorization is based on instructions provided in ISO 25063:2014 Common Industry Format (CIF) for usability: Context of use description. However, these subsections begin with a term “system-of-interest” that actually cannot be categorized as a contextual component.

3.3.1.1 *The system-of-interest*

This study adopts the concept *system-of-interest* agreed upon by several ISO standards, as “the system whose life cycle is under consideration in the context.” It is not a part of the context, but it is the centric subject for which the context is investigated. All the relevant components in the context influence the system-of-interest. For an implementation context, a system-of-interest is a modeling architecture with a goal of underpinning implementation of model driven system integration. The modeling architectures shall support implementation (in an implementation context) in addition to processes and information flows (in an application context). Unlike the traditional modeling architectures based on application contexts focusing on supporting information model development, this also supports information model use.

Hence, the subject domain of an architecture is the software-intensive system implementation, more specifically, for a system aiding engineering activities in the process of product realization. The generic information

models, the formal data exchange structure and the standardized data access interfaces defined within the modeling architecture of ISO 10303 STEP facilitate a primary part of the architecture.

3.3.1.2 *Users*

Technology enhances human capabilities to address human needs. When it comes to IT, the human capabilities need to be enhanced to take control over the technology in turn (Syväjärvi, et al., 2005). To take advantage of information models, three major types of human users of IT systems are involved: Application users, implementers and model developers. These three user groups are introduced and discussed in detail in the second section of the appended paper D.

This categorization is also valid in implementation contexts (Figure 3.8). The implementers are direct and primary users in the implementation contexts, initiate use of information models in implementation contexts. The implementers are concerned with the application users, as well as customers in Figure 3.8, during their implementation activities. The application users and the customers can be two adjacent players in a supply chain who need to exchange data represented in a data set, or can be two adjacent roles in a workflow in the same organization, e.g. a product designer and a process planner. The model developers should be aware of the implementation contexts, and are secondary users in the implementation context. It means that they may be applied for maintenance tasks.

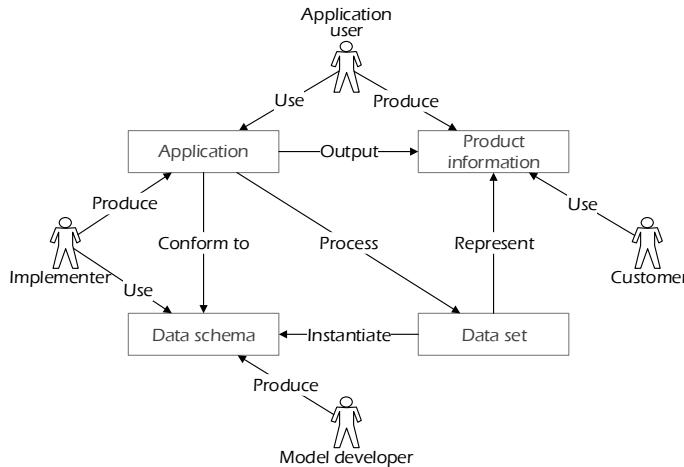


Figure 3.8 User groups and their positions in an implementation context.

This categorization is based on roles, but should not be applied to a specific person because a person can easily switch her/his role depending on the task the person is taking. For instance, most model developers are also industrial practitioners, who can be application users or customers. End-user developers (Lieberman, et al., 2006) are also a widely observed role in the world of manufacturing engineering, who can switch between the application users and the implementers in different scenarios.

3.3.1.3 Tasks

Tasks are represented by a series of interaction activities that can form a hierarchical structure (Annett and Duncan, 1967). Defining a task is essentially context dependent. Users' goals and available technologies often determine content of tasks. Taking CADCAM integration for example, there seems to be a kind of linear-distributed, phase-oriented allocation of tasks as displayed in Figure 3.9, but the truth is that a structured engineering process cannot guarantee a successful solution, while a flexible style is recommended by most experienced mechanical designers. In the appended paper B and E, the tasks in implementation contexts have been discussed in two system integration projects for cutting tool classification and kinematic data exchange respectively.

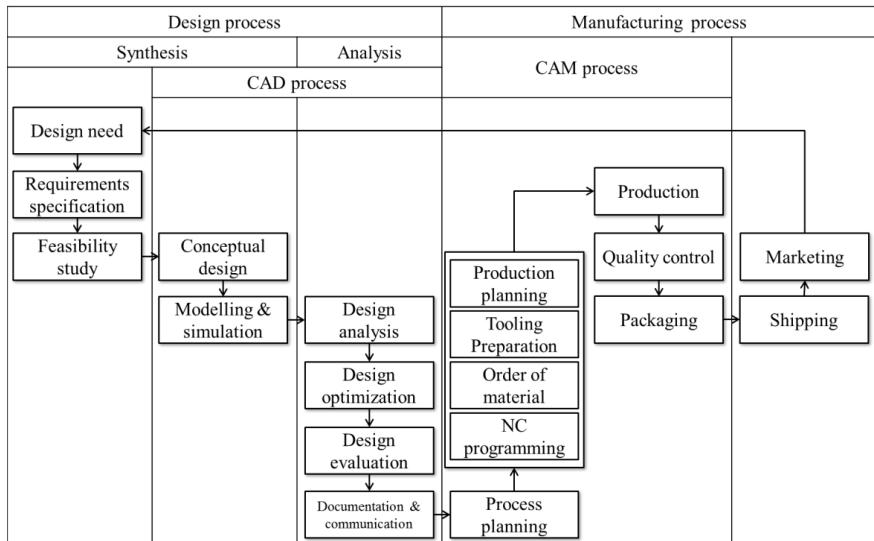


Figure 3.9 Tasks performed with CAD/CAM systems, based on (Makris, et al. 2014; Zeid, 1991).

Particularly for a context of engineering use of technologies, at a micro level, a pattern can be generalized with a cycle of analysis, synthesis and evaluation. Engineers in either software industry or manufacturing industry are used to shifting across these three modes and applications should provide support for these modes and the shifts. Analysis as a cognitive process requires presentation and visualization of knowledge in a structured way. Synthesis as mainly a motor process requires aids on intuitive integration and manipulation of information. Timely feedback is essential for evaluation. The shifts between these actions need seamless integration of functions and data, because most design breakthroughs happen during rapid shifting which is mostly influenced by cognitive activities of engineers. In general, information technology should provide sufficient support for the cycle and it is the highly extensible and portable standardized information models that lead to an opportunity to help engineers in different domains to complete this cycle effectively and efficiently.

3.3.1.4 *Technical environments*

The technologies that support system integration in an engineering context can be categorized as two types: Implementation platforms and information models. An implementation platform is usually a function-oriented package binding with a designated IT environment and with dedicated interfaces for implementers. In the third section of the appended paper D, a thorough discussion is made for the state-of-art implementation platform based on standardized information models.

It is the information model that establishes infrastructures for the implementation platforms and the implementation contexts. Semantic interoperability (independent of applications and implementation processes) is often a major goal to deliver standardized information models. The portability issue of the information models originates from this goal and is difficult to be tackled by normal implementation platforms that simply comply with the standardized data access interfaces, e.g. SDAI (Standard Data Access Interface, ISO, 1998a).

In an application context, information technologies are supposed to augment the human intellectual ability and produce artefacts accurately, quickly and easily. However, currently most engineers are “kidnaped” by technologies, i.e. in a tool-driven manner. The data they create, manipulate and communicate is limited by applications. Consequently, organizations in manufacturing industry drives employees, suppliers, IT supporters partially based on the software applications they are using. Hence, an important contribution of the model driven approach is to return the ownership of information back to engineers. By widely-accepted, vendor-independent, standardized modeling technologies, users are allowed to control their information assets in any way they prefer and to convey their knowledge (requirements, specifications, designs, reasoning, etc.) accurately with others.

3.3.2 Use of implementation contexts

Figure 3.10 illustrates position of the implementation context in the modeling process. This illustration is an update of Figure 3.4 due to the involvement of the implementation context, and still takes STEP as example. In

the illustration, the implementation context is used to define a set of information requirements for the use of a data schema. The meanings of this illustration can be obtained by row and column. Rows group specific processes and columns group similar levels of abstraction.

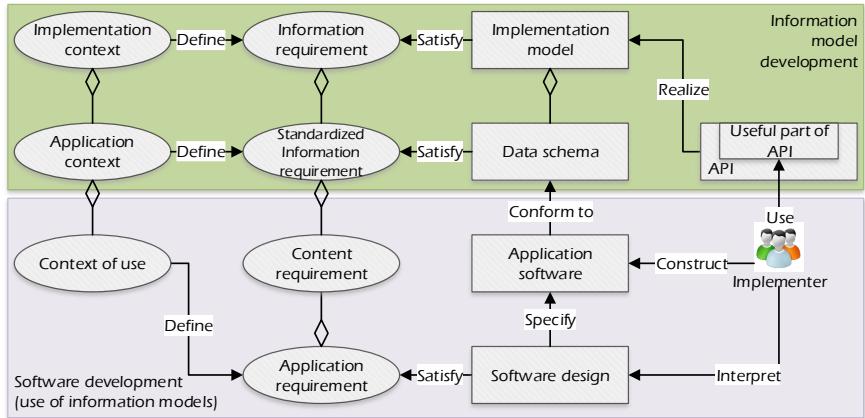


Figure 3.10 In the information model development process.

The original standardized model development process is simplified into the second row in Figure 3.10. Model artifacts less useful, e.g. AAM (Application Activity model) and ARM (Application Reference Model), are omitted only for simplification of illustration. To avoid confusion, the original information requirements are renamed as standardized information requirements in Figure 3.10. One result of this process is still the AIM (Application Interpreted Model) which is not recommended to be used by implementers directly. Instead, an implementation model encapsulates the AIM and is realized by an API for implementers. The API is also an encapsulation of the traditional interface techniques, i.e. SDAI and conformance testing tools. The process to create the implementation model and the API is the first row of Figure 3.10. Section 3.4 will introduce how to specify a new set of information requirements and the implementation models. Section 5.1 will use a case study to exemplify the creation of the API. The third row and the fourth row depict a software implementation process with artifacts that are relevant for this study.

Columns of Figure 3.10 represent groups with similar levels of abstraction. Contexts and requirements formulate the first and the second columns to demonstrate their imperative roles in any development process. Between the contexts and the requirements, the only type of relationship is aggregation. It suggests inclusive relationships, e.g. between the implementation context and the application context. Consequently, the following artifacts also have same relationship. Then, the third column represents artifacts satisfying the requirements.

Similar to any context, the usage of implementation context is to generate requirements. For information models, the set of requirements are still called information requirements. To develop a new modeling architecture, information requirements still describe 1) what should be standardized in a data schema; 2) what should be included to use the data schema during a use process (i.e. implementation). From a semiotic perspective, the former is at a semantic level, which can be facilitated in the original modeling architectures; the latter is at a pragmatic level and will be introduced in Section 3.4.3 for creation of an implementation model.

Software implementation process, implementer and implementation context

Implementers perform implementation based on an application context through a *software implementation process* “to produce a specific system element (software item) implemented in software.” (ISO/IEC/IEEE, 2015) During a software lifecycle, an information model can be shared, ported, or reused. The software implementation process, initiated by diverse technical needs and business needs, that results in heterogeneous forms of application software and in great complexity of software engineering practices.

The process is performed by *implementers* who are individuals or organizations that perform implementation tasks (ISO, 2008). The implementers may have different understanding in data, functions, contexts, etc., with business and technical constraints. This fact leads to differences in design, construction and deployment processes and the differences in delivered software functions and capabilities. All these differences make it not easy to create an implementation-oriented artifact.

This study introduces the term *implementation context* to define the environment where implementers develop software application using information models. In this thesis, the implementation context is delimited as the intended context of using an information model. However, the concept of “implementation context” can be generalized for any subject directly used by implementers, which is out of the scope of this thesis.

Both implementation contexts and application contexts will be translated to information requirements and information models. Differences between the products from the two contexts lie in detailing levels. The implementation context needs an information model specified in accordance with how it will be used, i.e. to support implementation, application, and maintenance. On the other hand, the application context needs an information model specified regardless of implementation details, but only for the use of applications, i.e. information flow and processing during engineering activities. Implementation contexts require less detailed semantics and more detailed pragmatics. Therefore, the produced information requirements and information models should represent these requirements.

3.3.3 Discussions of implementation contexts

There are several benefits by introducing implementation contexts for both information models developers and implementers. 1) The implementation contexts are an effective base for developing an information model friendly for its users, which boosts the pragmatic interoperability. 2) The software implementation process will not be entangled with the model development process. All relevant artifacts have been encapsulated in APIs as the only interface for implementers. Just like the construction stage in a normal implementation process, implementers choose APIs for different components according to software design. 3) Information model developers and implementers can reach a consensus on modularization of APIs by understanding implementation contexts. As introduced in Section 2.2.2, Implementers should select useful parts of data schemas for implementation. The modularization is a technique to facilitate the selection in a way more familiar to implementers. Exemplification of the modularization of APIs can be found in Section 5.1 and paper D.

ISO/IEC/IEEE (2011a) regarded implementation free as an important characteristic of individual requirements and this principle is not violated

here. Cooper, et al. (2014) considered the main principle for requirement definition is to “define what the product will do before you design how the product will do it.” Does the focus on implementation context diminish the quality of the information requirements? Note that the system-of-interest when defining information requirements is the information model, more specifically, the data schema. The information requirement is introduced to develop information models, rather than to be implemented in an application. Hence, a set of information requirements for an information models may address detailed information about application implementation (not model development) without hurting its implementation independency. The information requirements should be free from methods to develop an information model.

3.4 Implementation models

The *implementation model* is proposed as an information model that encapsulates one or more standardized data schemas, with additional supportive functions for data set manipulation, to satisfy information requirements in an implementation context.

3.4.1 Semiotics of information models

Creation of an implementation model requires an in-depth understanding of an implementation context (Figure 3.14), of which information models are actually the most confusing concept for all stakeholders. It is therefore critical to fully clarify this concept as a part of implementation and application. An information model is used for communication with a process of representation and interpretation, which is a “sign process”, i.e. “semiosis” defined by Peirce (1907). Human interpretation, i.e. pragmatic interoperability, has been identified as a gap to use information models. Hence, it is useful to define and categorize information models from the perspective of semiotics, which will lead to the exact problematic parts.

Data sets and data schemas are two information models mostly referred in this study and are in direct relation with stakeholders. They are mainly used by application users and implementers respectively. A data set conforming to a protocol can be roughly treated as an instance of a (part of) data schema of the protocol. For a normal application protocol in STEP,

three semiotic aspects of data sets and data schemas are facilitated by different specifications (Table 3.1). This semiotic perspective can lead to in-depth understanding of current problems and practical needs.

	Syntax	Semantics	Pragmatics
Data set	ISO 10303-21, ISO 10303-28, etc.	an AIM of an AP	UI specified by implementers
Data schema	ISO 10303-11 (EXPRESS)	AAM/ARM, etc.	AP documents, ISO 10303-22 (SDAI), ISO 10303-27, etc.

Table 3.1 Specifications to facilitate semiotic aspects of information models standardized in the framework of STEP.

Data sets are application-user-oriented. For example, a structurally complete STEP data set conforming to ISO 10303-21 is displayed in Table 3.2. The syntax of all the lines of Table 3.2 is governed by ISO 10303-21; this syntax of data set can be reformatted to XML governed ISO 10303-28 (ISO, 2007b). Semantics of a data set is specified by a data schema, i.e. each instance in a data set, exemplified in Table 3.2, instantiates an entity in the AIM of ISO 10303-242; the complete data set is valid according to the AIM. Pragmatics of a data set is ideally determined by UI of applications (and practically by implementers). UIs are designed to display different facets of a data set, according to designated tasks. For example, Table 3.2 shows an excerpt of the data set for the kinematic link. Users of Siemens NX can explicitly discern a kinematic link with its corresponding geometry (Figure 3.11) represented in the data set; users of STEP-NC Machine interpret the kinematic links with motion simulation (Figure 3.12); users may also interpret a data set in a STEP model translator without 3D visualization (Figure 3.13).

```
#2695=KINEMATIC_LINK('base');
#2696=RIGID_LINK REPRESENTATION('',( #2736, #2747 ), #2665, #2
695 );
#2697=KINEMATIC_LINK REPRESENTATION_ASSOCIATION(' ',' ', #26
96, #176 );
#2698=PRODUCT_DEFINITION_RELATIONSHIP_KINEMATICS(' ', $, #33
);
#2699=CONTEXT_DEPENDENT_KINEMATIC_LINK REPRESENTATION( #26
97, #2698 );
```

Table 3.2 An excerpt of a p21 (conforming to ISO 10303-21) data set.

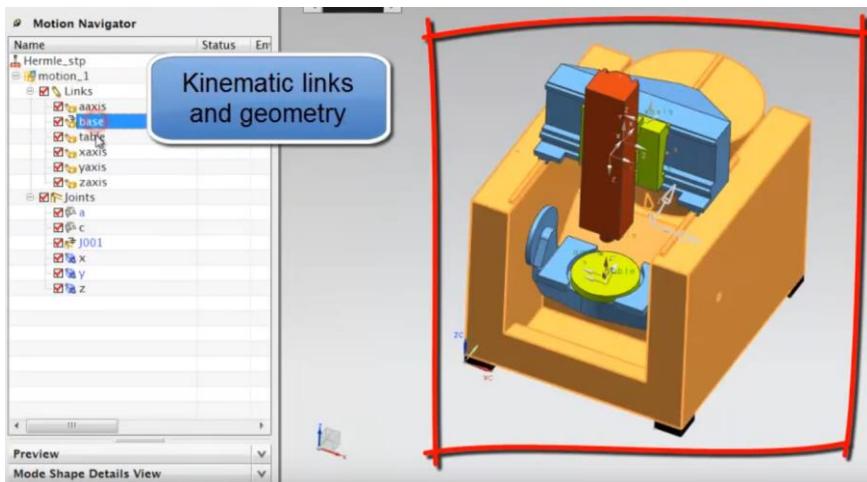


Figure 3.11 UI of Siemens NX 7.5 visualizes a kinematic link in relation to its corresponding geometry.

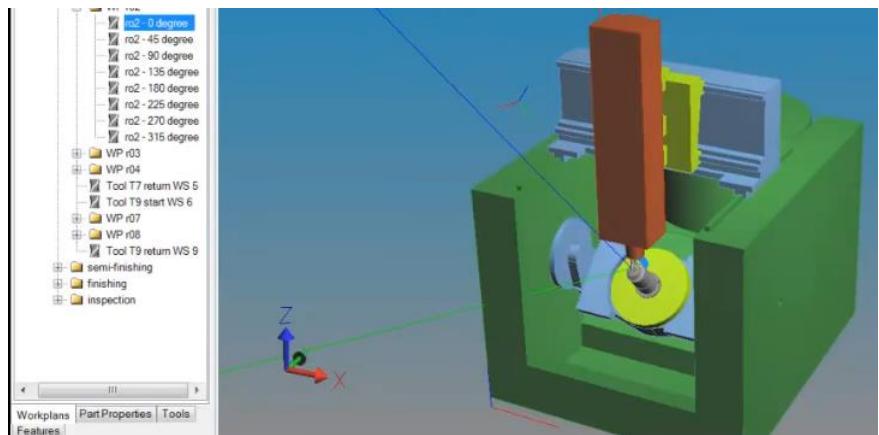


Figure 3.12 UI of STEP-NC Machine 9.38 visualizes motion simulation.

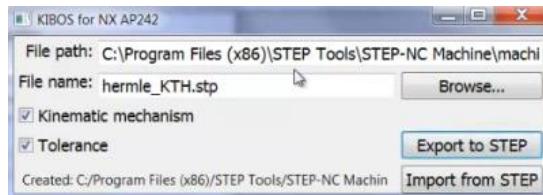


Figure 3.13 UI of a STEP AP242 model translator developed in this study.

As defined in Table 3.1, a STEP data schema (e.g. an AIM) also has its semiotic aspects facilitated with a different set of techniques. For example, the instances Table 3.2 are governed by the AP242 schema partly listed in Table 3.3, where the textual descriptions are based on ISO 10303-105:2014. For such kind of data schemas, ISO 10303 STEP defines EXPRESS as a modeling language to govern the syntax. The EXPRESS is also reused in other standards, e.g. ISO 13584 PLib, ISO 14649 STEP-NC and ISO 16739 IFC (Industry Foundation Classes). Semantics are governed by AAM and ARM that respectively specify application contexts and corresponding information requirements at different levels of abstraction. Based on AAM and ARM, model developers should designate meanings to the data schemas. This designation process for an AP is called “interpretation” (ISO, 1994). Pragmatically, a verbose “dictionary” (AP documents) and a “gram-

mar book” (SDAI and programming language bindings) are given to implementers to interpret the data schemas. These tools are insufficient because they are actually presented only at a semantic level and a syntactical level, respectively. These tools cannot ensure efficiency and effectiveness of human interpretation.

```
SCHEMA ap242_managed_model_based_3d_engineering_mim_lf;
```

```
...
```

```
(*
```

A context_dependent_kinematic_link_representation is the association of a kinematic_link_representation_association with a product_definition_relationship_kinematics. The kinematic_link_representation_association identifies the shape of a kinematic_link_representation as it plays the role of the related_product_definition in the product_definition_relationship.

EXPRESS specification:

```
*)
```

```
ENTITY context_dependent_kinematic_link_representation;
  representation_relation :  
    kinematic_link_representation_association;  
  represented_product_relation :  
    product_definition_relationship_kinematics;  
END_ENTITY;
```

```
(*
```

Attribute definitions:

representation_relation: a kinematic_link_representation_association that is associated with the product_definition_relationship_kinematics.

represented_product_relation: a product_definition_relationship_kinematics that identifies the shape of the related kinematic_link_representation_association in the context of a product_definition_relationship.

```
*)
```

```
...
```

```
(*
```

A kinematic_link is a type of vertex representation of the topological aspects associated with a rigid part of a mechanism.

EXPRESS specification:

```

*)
ENTITY kinematic_link
  SUBTYPE OF (vertex);
END_ENTITY;
...
(*
A kinematic_link_representation_association is a type of representation_relationship that associates a representation with a kinematic_link_representation.

NOTE A kinematic_link_representation_association is used to define the shape of a link by associating a shape_representation to the corresponding kinematic_link_representation.

EXPRESS specification:
*)
ENTITY kinematic_link_representation_association
  SUBTYPE OF (representation_relationship);
  SELF\representation_relationship.rep_1 :
kinematic_link_representation;
  SELF\representation_relationship.rep_2 :
shape_representation;
WHERE
  wr1:
( ( SELF\representation_relationship.rep_2.context_of_items
  := SELF\representation_relationship.rep_1.
context_of_items ) OR
( 'AP242_MANAGED_MODEL_BASED_3D_ENGINEERING_MIM_LF.' +
'REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION' IN
TYPEOF( SELF ) ) );
END_ENTITY;
(*
Attribute definitions:
rep_1: the kinematic_link_representation with which rep_2 is associated.
rep_2: the shape_representation with which rep_1 is associated.

Formal propositions:
WR1: The context_of_items of rep_2 shall be identical to the link frame of the
kinematic_link_representation with which rep_2 is associated.

```

```

*)
...
(*
A product_definition_relationship_kinematics is a type of property_definition.
The product_definition_relationship_kinematics specifies the kinematic prop-
erty of a product relationship.

EXPRESS specification:
*)
ENTITY product_definition_relationship_kinematics
  SUBTYPE OF (property_definition);
  SELF\property_definition.definition :
product_definition_relationship;
UNIQUE
  ur1: definition;
END_ENTITY;
(*
Attribute definitions:
Definition: an inherited attribute that shall be of type product_definition.

Formal propositions:
UR1: The definition shall be unique within a population of product_defini-
tion_relationship_kinematics.

*)
...
(*
A rigid_link_representation is a type of kinematic_link_representation.

EXPRESS specification:
*)
ENTITY rigid_link_representation
  SUBTYPE OF (kinematic_link_representation);
END_ENTITY;
...
END_SCHEMA; --
p242_managed_model_based_3d_engineering_mim_lf

```

Table 3.3 An example of an ISO 10303 data schema
(a part of the long-form schema of STEP AP242).

From the semiotic perspective, the aim of this study is to use implementation models in a modeling architecture to effectively facilitate pragmatics, i.e. human interpretation of data schemas. Data sets do not concern this study. The problems in terms of semiotics have been clarified previously: 1) Implementers (i.e. interpreters) have to generate semantics of data schemas, which should not be an interpreters' task and 2) the designated access interface (SDAI) of data schemas is not sufficient to interpret the data schemas. Hence, implementation models should satisfy 1) suitable abstraction of semantics for implementation and 2) specification to facilitate interpretation at a pragmatic level. These two requirements are underlying principles to define a new modeling architecture.

3.4.2 Encapsulation

As implied earlier, encapsulation is crucial for the creation of an implementation model, i.e. defining what should be exposed and not to the implementers. The nature of high semantic completeness and high extensibility of a data schema makes not all its elements useful for a particular implementation. Thereby, data schemas are often tailored in practice because implementing an entire application protocol is usually not necessary. Only those relevant for implementation shall be exposed in the implementation models. Moreover, entities in data schemas only concerned about application domains are not sufficient for implementation, i.e. a lack of functionalities.

As defined previously, these facts are managed by an implementation model with encapsulation is performed in three steps: 1) Completeness and granularity of existing verbose data schemas are properly managed and abstracted; 2) additional functionalities to complete implementation tasks are included; 3) entities and functionalities are communicated in a pragmatically effective manner. The derived new set of information requirements should reflect the first two steps of this encapsulation anyhow. The third step is about pragmatic use of an implementation model, i.e. application programming interfaces (APIs) should be realized.

It is still important to keep the existing modeling architecture and the existing implementation process based on the application contexts. The encapsulation make some content in data schemas invisible but does not abandon or replace the invisible content. This means that this solution

should provide a wide range of abstraction levels to facilitate different users' needs in implementation contexts. In particular, there should be a mechanism to bypass the encapsulation and go back to the original way. This mechanism is presented as a layered architecture for system integration in Section 3.5. There are several benefits from the mechanism: 1) Implementation models is not compulsory and can be developed for a part of a data schema. 2) Existing implementers are also free to reuse their knowledge to implement the traditional information models. 3) Implementation projects should be allowed to switch implementation styles between the new one and the old one to meet the needs of different implementers.

3.4.3 Information requirements

The creation of an implementation model will be presented in the next subsection, but as a critical step, the creation of information requirements is presented here in a separate subsection. An implementation context is an initial point for the creation of an implementation model; the first step to create an implementation model is the creation of a set of information requirements based on the implementation context. As described in Section 3.3, implementers, applications and application contexts are primary components of an implementation context, which can be regrouped and connected in Figure 3.14. As shown in this illustration, application users and implementers are considered as the most concerned stakeholders; implementers should develop integration solutions to integrate existing applications, probably by creating new applications to facilitate the integration; data sets and data schemas, either standardized or vendor-defined, are also components in concern to represent domain knowledge. The use of these contextual components (including domain knowledge) will be introduced to ensure the pragmatics of information requirements (Figure 3.15).

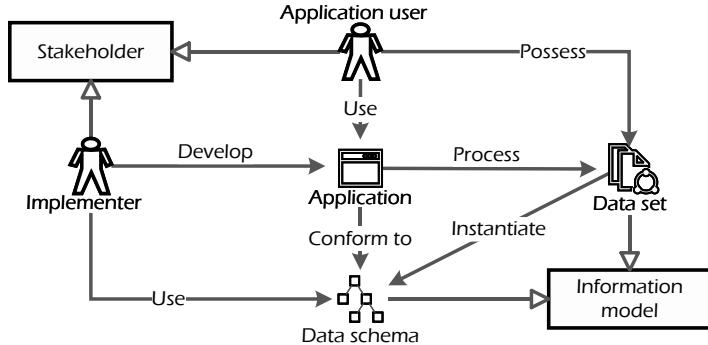


Figure 3.14 Components in an implementation context.

An important issue in STEP modeling architecture is that information requirements are not designed to be interpreted by computers, but rather by human beings. Information requirements are the main leverage to create (for model developers) and to use (for implementers) information models in a modeling architecture at a semantic level. Properly composed, information requirements are useful for both model developers and implementers. Model developers use them to generate implementation models. Implementers use them as a reference directly describing the implementation models. Hence, all semiotic aspects of information requirements should serve a purpose of effective human interpretation.

Syntactically, the information requirements should be expressed in a form familiar to implementers. UML (Unified Modeling Language) may today be the graphical modeling standard that is most familiar to implementers. Therefore, this study describes the information requirements in a form of simplified UML class diagrams (Rumbaugh, et al., 2004). The simplification is made to increase readability by omitting elements either obvious or redundant: Class compartments, indicators for abstract class, ease of navigation for aggregation, multiplicity and visibility. Figure 3.14 and Figure 3.15 exemplify this simplification style. This simplification does not violate completeness but increases readability. To sum up, the syntax is a graph with nodes and edges, decorated by arrows defined in UML. Note that in future there might be another solution replacing UML, and the encapsulated data schemas remains unchanged.

Semantic aspects of information requirements are limited, because computer interpretation is not required. Still, semantics of information requirements will greatly determine semantics of implementation models. Hence, the semantics of information requirements should be communicated at a suitable degree of granularity, completeness and validity. Thus, a controlled vocabulary is demanded. This mission is almost the same as creating an application ontology: Both tasks and domains should be depended on (Guarino, 1998). This study selects a series of principles (Table 3.4) for ontology design defined by Arp, et al. (2015), which are usable for representing information requirements.

Title	Description
Realism	Represent only those in reality.
Perspectivalism	Do not seek to represent entire reality. A modular approach is promoted.
Fallibilism	Change is possible, trackable and open to users.
Open-world	Allow continuous extension and amendment to adjust broadness and granularity.
Adequatism	Allow for multiple levels of granularity to adapt needs in multiple domains.
Relevance	Include terms used by relevant systems.
Consensus	Strive to ensure maximal consensus among stakeholders.
Overlap identification	For reuse between modules, identify synonyms, terms with different meanings, terms used at different levels of granularity.
Singularity	Use singular nouns for nodes.
Lowercase	Try to avoid uppercase and abbreviation.
Univocity	Hold exactly one meaning for each term (node) and each relation (edge).
Avoiding mass	Represent only countable terms.
Universalism	Represent only what is general.

Table 3.4 Principles usable for representing information requirements.

Pragmatically, information requirements should be effectively interpreted by both model developers and implementers. Information requirements from an implementation context is essentially a compromise among relevant contextual components: Knowledge of relevant domains, integrated applications and implementers (Figure 3.15). This compromise should be properly documented and tracked. 1) Domain knowledge is a critical pragmatic part of any implementation practice. The existing data schemas are considered as the most important source for the domain semantics, because the direct implementation goal is system integration conforming to the data schemas. Moreover, a formalized data schema is supposed to be highly reliable compared with knowledge of application users. 2) As shown in Figure 3.15, the information requirements should also address concerns of implementation. That is, information requirements should resolve foreseeable mismatches between data schemas and common applications. For instance, interfaces of applications may have different preferences on either global or local coordinates. 3) For implementers, the most important pragmatic issue is in relation to programming languages and should be directly reflected by realization techniques of the implementation models.

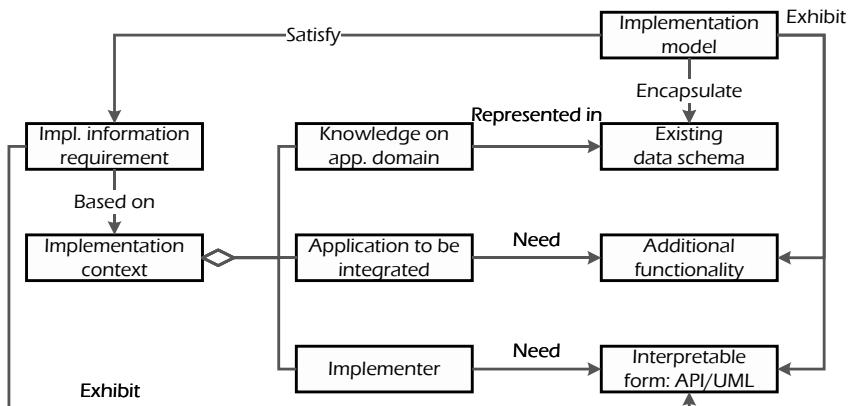


Figure 3.15 Information requirements based on implementation contexts.

3.4.4 Creation of an implementation model

As defined earlier, an implementation model is also an information model, with its specialized semiotics (Table 3.5). An implementation model is recommended to be realized in an API. Hence, syntax should be governed by specific programming languages. Semantics of implementation model is largely determined by information requirements. The mapping between the information requirements to an implementation model should be consistent and clear, i.e. each class and each relation in-between in the information requirements should have corresponding elements with (almost) same naming in the implementation models; thus, the information requirements in simplified UML class diagrams can be a document to brief the semantic structure of the API to implementers. Pragmatically, an implementation model should be interpreted by implementers with its API documents.

	Syntax	Semantics	Pragmatics
Implementation model	Programming language, e.g. Java	Information requirements	API document

Table 3.5 Specifications facilitating semiotics of implementation models.

With semantics about application domains defined by information requirements, the remaining task it to implement the information requirements in a form of API with proper documentation. Thus, properties, methods and algorithms should be completed based on the information requirements. This is not a study focusing on software engineering; therefore, the art of designing a good API is not investigated in detail. However, there are still some noticeable fundamental principles (those principles explicitly regarding definition of semantics and syntax are excluded):

- Documentation matters. Any API is all about reuse; the reuse, even for components with best design, cannot happen without proper documentation (Parnas, 1994).
- Document before coding. Thus, documents are not affected by coding (Henning, 2009).
- Documents “read like a book”. Documentation should be kept as simple, intuitive and elegant as possible (Jacobson, et al., 2011).

- “Smaller is better”. The fewer the elements, the easier it is to interpret, learn and use; adding any non-fundamental functions should be double checked (Henning, 2009). For an API in use, “you can always add, but you can never remove” (Bloch, 2006).
- “Agility trumps completeness”. Start API with specification as short as possible, so that it is easy to read and maintain (Bloch, 2006).
- Dictate semantics (Henning, 2009). What have been defined by information requirements should be obeyed throughout the entire content of the API.
- Keep consistency. For such a complex (interdisciplinary) API, ergonomics depends on consistency which should always be maintained to ease use, memorizing and transference of learning (Henning, 2009).
- Moderate impacts of implementation. As always, API is implementation-oriented, but cannot be over-specified (Bloch, 2006), i.e. impacted by implementation details too much.
- Demonstration matters. A live demonstrable case can directly point to its value for users. Thereby, it is important to select significant use cases and working on sample codes for them (Jacobson, et al., 2011).

Note that, with the help of implementation models, implementers may still feel heavy loaded because knowledge about relevant domains seems too much. Figure 3.15 shows that implementers are supposed to possess knowledge in both domains, in software engineering as well as in production engineering. Generally, the assumption can be true because requirement engineering and system architecting help to bridge the domains. However, for professional system development, the requirement engineering and the system architecting can still be a huge workload. An implementation model can relax implementers from knowledge about information modeling, but cannot relax them from knowledge about the application domain. Chapter 4 aims at the latter relaxation.

3.4.5 An exemplification

In the appended paper D, the STEP toolbox API is an example to realize an implementation model. In this example, objectives are to simplify the traditional implementation process and to reduce learning effort of the implementers. This is exactly why an implementation model is incorporated. The implementation model is used to increases the pragmatic interoperability of an information model. Although not explicitly stated in the paper D, the STEP toolbox is developed based on the implementation context, so that its pragmatic interoperability can be ensured.

Kinematic data exchange is a target to be implemented with a module in the STEP Toolbox used in paper D. Implementers are supposed to use the STEP Toolbox API to operate data sets directly. Therefore, experienced model developers can begin the creation of the implementation models with analysis of exemplified instantiation. For instance, the following excerpt (Figure 3.16) represents description of a kinematic link and its relation with product definition. In paper D, there are also excerpts for a kinematic topologic structure and relation with geometric information.

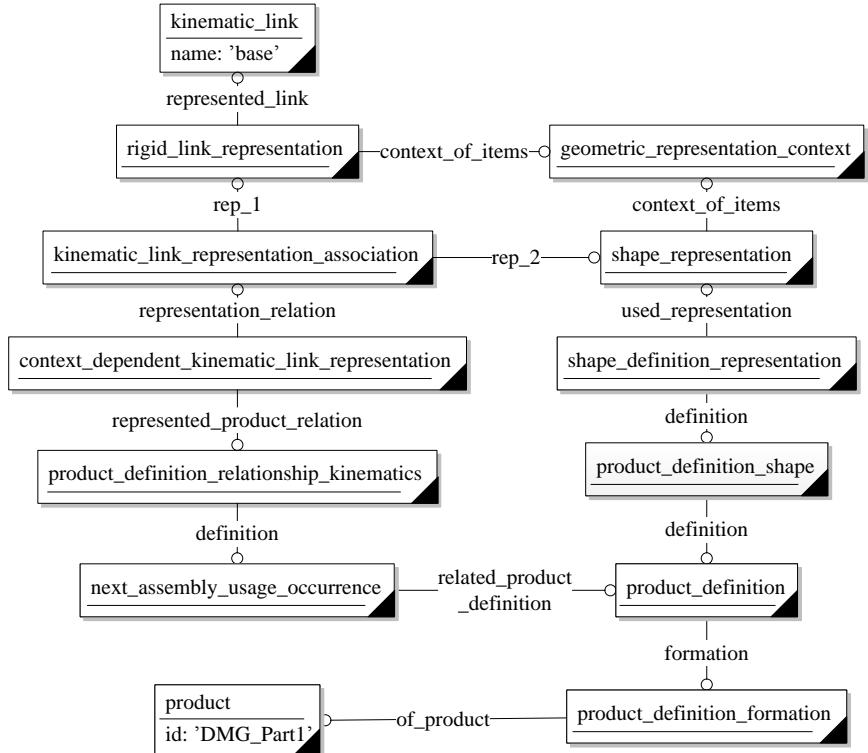


Figure 3.16 An example for instantiating a kinematic link, excerpted from an AP242 data set.

A valid sample data set (e.g. Figure 3.16) may be a suitable start point for implementers to understand a standard, because this is an expected output, but it is not enough for developing an implementation model. An implementation model should support a reasonably complete scope for basic I/O operations. Its success is determined by success of implemented applications to interpret and process possible data sets. This interpretation process regarding data sets is the main focus of system integration and the reason why information models are needed. The implementation models should help implementers to use the data schemas rather than to process a limited set of data sets.

Based on the analysis of the data schema and other components in the implementation context, model developers can elicit information requirements, diagrammatically presented in Figure 3.17. The information requirements are presented here in a simplified form of the UML class diagram which is understandable by implementers. This illustration includes simplification that temporarily omits attributes and basic functions. As illustrated, nodes and edges are used to consolidate complex data structures in the data schemas. What has been hidden is difficult for implementers but easier for model developers. Hence, the complexity does not disappear, but is managed by more suitable experts.

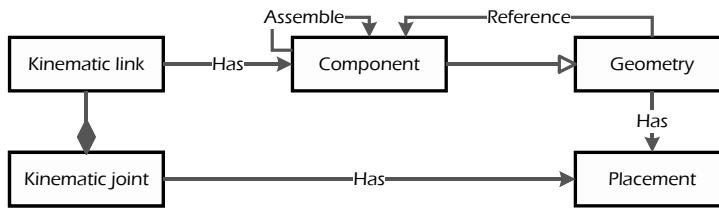


Figure 3.17 Diagrammatic information requirements for an implementation model for kinematic data exchange.

The implementation models should reflect all the elements with additional consideration about the integrated systems. A major concern about the systems in this case is about translation of coordinate systems. In AP242, placements of joints are coded in links' local coordinate systems, but these placement values may not be suitable for some applications only supporting a global manner for all inputs. In this example, not only is a general utility for coordinate system translation provided, coordinates for all kinematic elements in both manners are also provided explicitly. The result is a UML class diagram with more complete specification on properties (Figure 3.18).

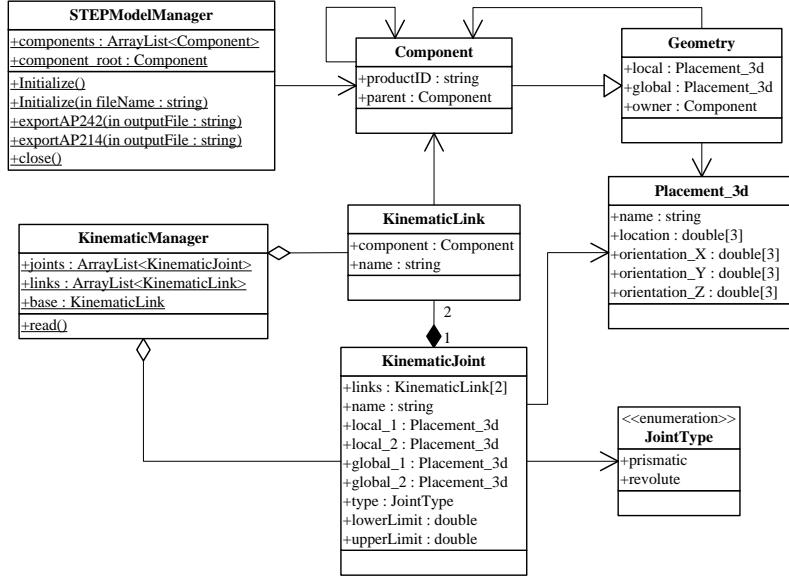


Figure 3.18 A class diagram for kinematics in the STEP Toolbox API for Java.

3.5 Layered architectures

Section 3.1 and 3.2 form the gap of this study with identification of the root cause. The gap leads to the introduction of the implementation contexts (Section 3.3) and the implementation models (Section 3.4). Nevertheless, the previous sections focused on what these new concepts were and how to develop them, but how to use them was rarely reached. Hence, the introduction of a modeling architecture is needed to make use of the new concepts in real business circumstances. This study does not define a specific solution of architectural design for application software. Instead, a modeling architecture is put forth as an infrastructure to facilitate system integration implementation toward maximum pragmatic interoperability.

There are actually noticeable gaps in implementation practices, as reasons why a new modeling architecture is needed in an implementation context: 1) Standardized information models are partly incompatible with a context of use of application software, due to a limited set of information requirements. As a result, function design of some applications was limited

by demanded compliance with standards (Lee, 1999). This fact indirectly hurts the supposedly implementation independency of standards. 2) There is a lack of inter-protocol solution in existing modeling architectures. An explicit geometric context to increase semantic coherency is valuable for communicating design intent (Hedlind and Kjellberg, 2015). This can lead to heavy workloads for implementers using protocols without explicit geometric representation. 3) In manufacturing industry, there is essentially a lack of resources for an interdisciplinary errand like implementing information standards. When an information model is designed to be kept away from the implementation context, the learning curve to implement an unfamiliar information standards becomes steep.

Hence, a modeling architecture is needed to provide implementation-oriented support, taking advantage of the proposed implementation models. The core part of the modeling architecture is an implementation model as a contextual layer illustrated in Figure 3.19. It is designed for integrated applications (as in the top layer of Figure 3.19) interacting with data sets (the bottom layer of Figure 3.19). The composable layer is to take advantage of the original STEP modeling architecture that guarantees development of good information standards. The contextual layer, is to bring portability into the consideration to reduce complexity in implementation. As discussed in Section 2.3, the layered design was to compensate the traditional composable solution by a contextual solution that CAx system implementers are more familiar with.

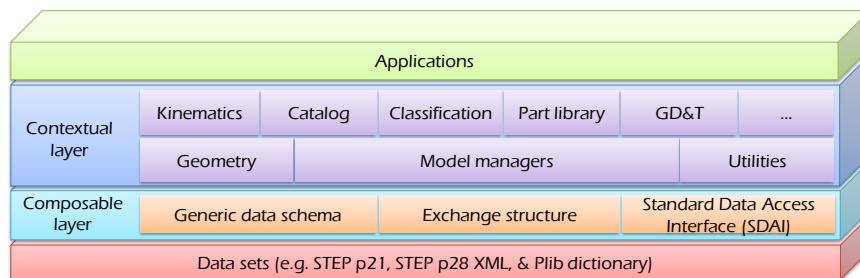


Figure 3.19 The layered modeling architecture.

3.5.1 The composable layer

For the composable layer, generic data schemas are adopted which can be instantiated in specific application contexts, such as STEP AP242 (ISO, 2014b) or STEP AP214 (ISO, 2010a). The generic data schemas are adopted due to the comprehensive semantic representation and high extensibility that are exhibited in different aspects. 1) For starters, these data schemas are applicable with existing useful implementation methods, e.g. SDAI and its bindings to common programming languages. Apart from SDAI, definitions of validation methods and exchange structures (such as XML representation and binary representation) constitute solid foundation for implementation and use. 2) A generic data schema usually has a large scope with a regular incrementing plan to facilitate a complete support for a large range of industrial sectors. For example, STEP AP242, initiated as a convergent application protocol to harmonize AP214 and AP203, aims at satisfying interoperability requirements from automotive, aerospace and defense industry (ASD, 2009). 3) The comprehensive infrastructure and the large scope make generic information models easy to extend to new application domains. A plenty of examples can be observed during development and application projects. Chen, et al. (2011) confirmed that STEP AP214 was the only standard able to represent all needed modules for factory layout, although complexity and efficiency were still highlighted issues. Hedlind and Kjellberg (2014) proposed extensions based on STEP p105 ed2 (ISO, 2014a) to explicitly represent function requirements on kinematic errors, which could be easily integrated into AP242 as well. Lundgren, et al. (2016) proposed a necessary extension for STEP AP242 and STEP-NC (ISO 14649) to enable system integration for process planning and quality assurance.

With the generic data schemas as a prerequisite, the composable layer represents the default implementation framework specified by ISO 10303. This framework is built in a typical composable way but inadequate for implementers, of which the semiotic aspects was discussed in Section 3.4. The provided implementation methods have no clue about any semantics, i.e. all the useful utilities are at a syntactic level. Hence, to use the existing implementation methods, a large chunk of code is needed to populate the semantics. Moreover, even an executable program with SDAI cannot guarantee a valid data set without validation. Validity of a data set depends on

the instantiated data schema. Besides EXPRESS, SDAI and validation, there are many specific concepts regarding modeling that should be comprehended. Some of the concepts may have little connection with either the software engineering domain or the manufacturing engineering domain, which are only related to information modeling. As a result, this layer has low initial usability for novice developers or in early iterations, but has high final usability for experienced developers or, sometimes, in late iterations.

3.5.2 The contextual layer

The contextual layer, i.e. the implementation model (Section 3.4), consists of components highly related to implementation details. It is designed to be usable for implementers as the primary users of the information models. This layer is directly affected by an implementation context and, therefore, only works with corresponding implementation methods. For example, Java is chosen as the major programming language in paper D to implement the layer, according to technical requirements of related projects. In addition, the application context should be also reflected in this layer, since the implementation context is greatly dependent on the application context which is mostly concerned by implementers.

Programming interfaces, e.g. written in Java language, are composed and packaged directly related with domain semantic concepts and relevant functions. In this way, the contextual layer facilitates specific information requirements as well as functional requirements elicited from an implementation context. Moreover, references of the interfaces should be presented in an implementer-friendly way, i.e. adopting semantics widely accepted in the manufacturing industry and the IT industry.

As stated in Section 2.3, the contextual approach has several advantages which make it usable for the implementation context. 1) The contextual layer has a high findability, where specific components are easy to locate for specific requirements. The components greatly related with the contexts are easy to learn and fast to use by the implementers with little knowledge about information modeling or with limited sources to perform formal regular model mappings for implementation. 2) The contextual layer provides “out of the box” solutions for specific implementation contexts. The implementers are provided with efficient ways to realize semantic concepts with predefined default values. Commonly used data sets like

p21 data sets can be interpreted, created, or processed directly in an efficient but still valid way. The layer can even provide extremely contextual solutions that are only suitable for integration with a specific CAx system.

3) From guidance tutorials to sample codes, the contextual layer should deliver sufficient knowledge support for the implementers to implement frequently used processes. Most implementers involved in projects related to the study had limited knowledge in either specific information models or general information modeling principles. Skipping most parts of manual interpretation of the data schemas can benefit the implementation targeted on specific contextual needs with a high degree of efficiency and error prevention. 4) The contextual solution with high efficiency and high learnability in specific contexts will motivate the implementers, especially the end-user programmers, to use the model driven solution more often.

3.5.3 Closed layered and open layered architectures

The overall layered structure of the architecture implies dependency from top to bottom, which has two ways to apply the architecture for implementation: Closed layered and open layered (Matha, 2008).

Contextual use of the architecture is closed layered and one-directional (Figure 3.20), where the developed applications only interact with the contextual layer, but do not talk to any other layer; the contextual layer only interacts with the composable layer but does not have any idea about the application built upon it. The contextual use is suitable for novice implementers or implementation projects that can be satisfactorily supported by the contextual layer. Novice implementers do not have to interact with the composable layers, because closed dependency between layers is regulated. A fully closed way to use the architecture is a violation of the encapsulation principle, which may decrease reusability and maintainability of applications with dependencies of more than one layer (Martin, 1996).

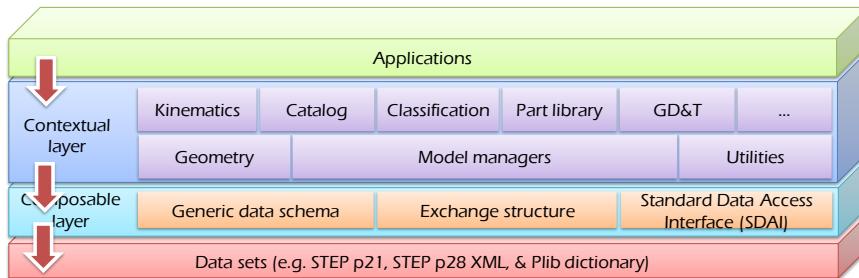


Figure 3.20 A closed layered architecture.

On the other hand, composable use of the architecture is one-directional but open layered (Figure 3.21), where the developed application can interact with both the contextual layer and the composable layer. That is, the contextual layer is “opened”. The composable use is suitable for expert implementers or implementation projects that cannot be fully supported solely by the contextual layer. This open channel from the contextual layer to the composable layer also helps novice implementers understand the composable layer as a generic solution based on a generic application context, which makes the learning process more efficient.

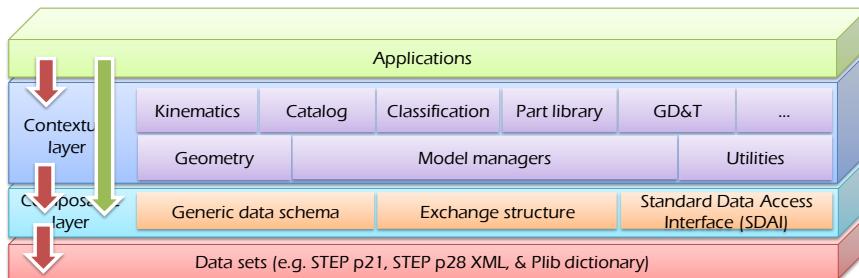


Figure 3.21 An open layered architecture.

Hence, the architecture should provide valid results with each of the single layer or with a combination of both layers. Expert implementers should be allowed to switch between both layers freely even for one data set. The open layered architecture should also be open enough so that the implementers can reach composable components beneath any contextual components for deep understanding of the implementation model.

This layered design aims at a portable solution for model driven system integration, with reduced complexity, increased efficiency and better learnability, compared with the conventional composable approach. A complete solution for all contexts is not the goal of the architecture. After all, aiming at all contexts means no concerned context at all. Instead, this study delivers architecting principles to make information models efficiently used in the implementation context with the aim at full interoperability. During the course of this study, this architecture has been refined, used and reused internally and externally in several projects. Concepts and functions are shared among the projects with similar or overlapped application contexts, so that contextual components at a proper level of abstraction built for one project can be efficiently reused by another project. In this way, the architecture, as a methodologic concept, helps engineers to reuse previous knowledge for problem solving (Spinellis and Gousios, 2009).

3.6 Discussions

The proposed modeling architecture can be seen as an enabling system (ISO/IEC/IEEE, 2015) to serve system integration using information models. It benefits the implementation context in at least three aspects, i.e. a high level of architectural agility, a high level of component decoupling and standardization of contractors.

The engineering context is essentially a changing environment and system integration is an important technique to cope with changes. The challenges are faced by IT developers as well as IT users. For long-time application in use, many factors of computerized information systems are constantly changing. From a technological perspective, hardware platforms, software frameworks and engineering systems are likely to be outdated or replaced at any time; implementers often need to switch programming languages or patterns to cope with the technical changes accordingly. From a business perspective, fast time to market (TTM) is an eternal demand, which puts engineering tools under continuous review for changing or upgrading; besides, potential changes on computerized information systems may also be triggered by new market demands, new regulations and new enterprise structures.

The changing environment demands an ability to make rapid responses by involved systems, i.e. high agility. The proposed modeling architecture can support effective and efficient implementation by reusable contextual solutions complementing the extensible composable information models. Thus, system integration can be implemented in accordance with unpredictable changes in an agile manner. Prototyping is easier to be implemented with such an agile infrastructure as the new modeling architecture.

System integration is always an ongoing project with possibilities of integrating new systems and disconnecting existing systems constantly. This demands ease to couple and decouple systems. Moreover, just like common implementation projects, dependency reduction is a core idea to achieve a high agility. The modeling architecture provides a solid basis for the dependency reduction. As illustrated in the upper scenario of Figure 3.22, with four exemplified systems communicating with each other, any change made in one of the four systems may cause at most three changes of communication mechanisms; a new additional system needs to establish four communication mechanisms in order to be integrated. An independent contractor reduces the dependency of integrated systems, as a leverage to replace the tedious communication mechanisms (the lower scenario of Figure 3.22). Information models acting as the contractor encapsulate all interfaces. Hence, only one change of integration topology is needed for coupling and decoupling of any system. The new topology isolates each system and enables each system to evolve independently, which greatly reduces coordination effort and increases implementation efficiency.

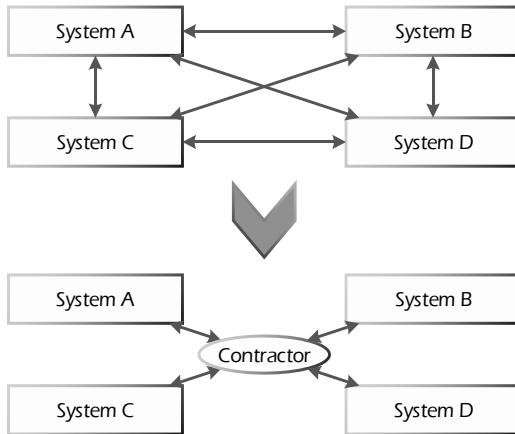


Figure 3.22 Decoupling to reduce dependency.

There are also potential downsides of introducing a contractor, e.g. increased complexity as a new component, decreased system performance with one more step for communication and a huge independency issue if the contractor is changed. Even though all the downsides are potential, these downsides contribute to a large part of a trade-off when developers need to decide whether to introduce a contractor and how to choose a suitable type of the contractor.

Standardization, as a major leverage of the proposed modeling architecture, is also a major feature of the contractors to deal with the above mentioned downsides. A well accepted standard often implies that experienced developers have already taken implementation complexity and performance into consideration. A mature standard suggests low changing possibility or at least backward compatibility. Besides, when using standards, there is usually a large resource pool for teambuilding and consulting and a better chance to reduce workloads for integration with other systems (which may be compatible with the same standard). Leveraging standards helps developers to adopt best practices instantly and to gain a better chance to be understood externally with a common framework and terminology. To sum up, standardization provides a usable, stable, and reliable infrastructure to optimize design for system integration.

Chapter 4

A model driven implementation process

Design is not just what it looks like and feels like.

Design is how it works.

- Steve Jobs

This chapter is about how to use a developed standardized generic information models. A magnificent standardization effort in production engineering is ISO 10303 STEP and the related standards (more details in Section 2.2.2). Modeling architectures of these standards are major references to discuss behaviors and structures of information models in this chapter.

These standardized information models are commonly treated as a technical constraint (Section 4.1.1) during implementation. However, their usages could be beyond this role (Section 4.1.2), because they are also notable for 1) focused concern with high semantic interoperability and 2) being used with a procedure similar to conventional model based software implementation processes. Hence, the standardized generic information models are potential to underpin the software implementation process (Section 4.1.3).

To fully take advantage of this potential, it is necessary to understand 1) what activities of the software implementation processes can be supported (Section 4.2), 2) what deliveries of the activities correspond to the information models in sequence or abstraction levels (Section 4.3) and 3) how to use the information models within the activities of the software implementation process (Section 4.4). The activities are grouped into five stages (in Section 4.5 through Section 4.9) and are underpinned by artifacts of the modeling architectures. The activities focus on delivering architectural design decisions to help industrial practitioners to rapidly generate feasible system integration solutions.

4.1 How to use information models

Today, information models are used in a limited way in practice in software engineering. Toward system integration for product realization, implementation is characterized by rich content and intensive human-model interaction; it is inevitable to use standardized generic information models to facilitate high interoperability in this context.

4.1.1 Current limited use of information models

To use an information model, typically implementers necessarily follow a manual mapping process (Figure 4.1), to understand and integrate a data schema. In this mapping, activity models (e.g. AAM in ISO 10303) and reference models (e.g. ARM in ISO 10303) are intermediate models to bridge interoperability needs in reality with a computer interpretable data schema (e.g. AIM in ISO 10303). This mapping process is defined in modeling architectures aiming at facilitating valid use of the information standards (more details in Section 2.2.2). Without the mapping process, it would be frustrating for anyone to locate right parts of a data schema to represent what should be communicated in a specific application context.

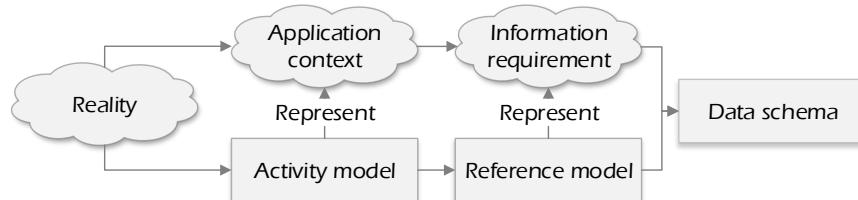


Figure 4.1 A traditional mapping process for using a data schemas.

If strictly obeying a modeling architecture (e.g. for ISO 10303), manually following this mapping can be a heavy burden on implementers. This fact violates a general intention of standardization, i.e. to reduce implementation burden (Lewis, et al., 2008). Particularly, implementers need to interpret the intermediate models such as activity models and reference models (Figure 4.1). As a result, modeling architectures are suitable for semantic interoperability and computer interpretation, but not suitable for pragmatic interoperability and human interpretation.

Now, zoom out to see the bigger picture, i.e. implementing system integration in an engineering context, where the use of the standards is a

portion of implementers' work: They map needs for communication (i.e. information requirements) to computer-interpretable data schemas (the middle column in Figure 4.2). Besides using the data schemas, implementers map other needs (e.g. independent content, independent functions and qualities) to an application. The mapping from reality to application is the mainstream in implementation (the left column in Figure 4.2), even without information models.

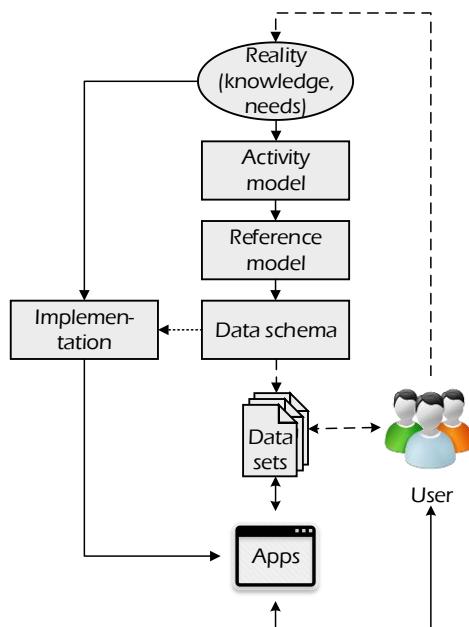


Figure 4.2 Current integration of the information model mapping with the implementation process.

These two processes, the information model mapping and the implementation process, are usually independent until the data schemas have been delivered as a technical constraint of construction. It means data schemas are commonly used as a standard to strictly follow. The intention of this use of standardization is right, i.e. reuse of existing product information and generating reusable product information. At most time, implementer designers select standards in early stages, e.g. requirement analysis,

but use them only in late stages, e.g. programming and testing. This situation, using standards as technical constraints in late stages, is observable when aiming at syntactical interoperability, e.g. using HTML, IMAP and TCP/IP. The standards regulate syntax for integration, which sounds familiar for implementers.

4.1.2 Envisioned use of information models

However, the current limited use of information models (Section 4.1.1) is a waste of talent and knowledge. Beyond regulating syntax, the standardized information models (e.g. ISO 10303) are characterized by the focus on interoperability for rich semantics. All information captured in the information models should be formally self-descriptive, to guarantee full semantic interoperability and computer interpretation. Usually, this is creation of top experts in subject domains. The knowledge within the information models is potentially highly useful for almost all stages of an implementation process, rather than just the construction stage.

In particular, system integration in this subject domain is challenged by integrating information architecture with software architecture. The involved context could be expanded to a large business domain (design, manufacturing, and resources) and a large technical domain (client application and server applications), as stated by Rosén (2010). This integration is characterized by complicated content-rich information flows and processes. This context has been formally represented by many artifacts in modeling architectures, such as the activity models and the reference models.

To sum up, information models possess some unique features in the standardization world, i.e. rich semantics on domain knowledge, creation of top domain experts, used in content-rich system integration, and needs to support computer interpretation at a semantic level. These facts lead to troubles and opportunities when using an information model. Both Section 3.2 and Section 4.1.1 discussed the troubles. To solve the troubles, Chapter 3 changed modeling architectures to support implementation; this chapter changes the way to use information models during implementation.

To achieve efficient and successful implementation, this study presents an enhanced implementation process closely integrating the manual interpretation of the model artifacts (Figure 4.3). In the enhanced implementation process, the standardized information models can potentially

support implementers in forms of comprehensive domain-specific knowledge. Meanwhile, it is possible to ease the manual mapping with early awareness of the standards in an implementation process. For instance, the activity models and the reference models can greatly benefit user study and requirement analysis. Consequently, the information models become support of implementation rather than just as technical constraints.

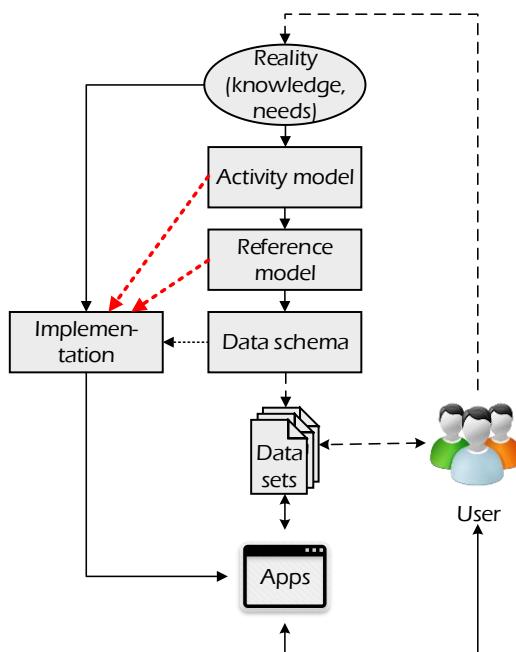


Figure 4.3 Closely integrating the mapping process with the implementation process.

4.1.3 Development of an integrated process for the envisioned use of information models

Implementation traditionally progress within a structured process with stages. Where, when and how to integrate the information models with the progression should be specified. This is crucial since most developers have limited knowledge and skills in information models, subject domains, or

software engineering. The following parts will reason how to use the information models in the implementation process with stages.

Essentially, the implementation process is also a mapping process from needs to accomplish goals or experiences in reality to a system satisfying this reality, with a process from context research to construction. This process is similar to the model mapping process. Thus, it will be helpful to examine the mapping processes in different domains and to discern similar stages that can support each other. This examination can start with a generalized mapping pattern faced by today's engineers, i.e. from complex concrete entities in reality to another type of complex concrete entities in reality (Figure 4.4) which are realized products (including digital products). For instance, software engineers translate users' needs to developed application software; manufacturing engineering translate business needs and customers' needs to physical products during a product realization process.



Figure 4.4 A general mapping pattern of engineering activities.

Models are adopted as an abstraction of concerned facts to translate the reality to the products (Figure 4.5). By definition, a model is usually an abstraction of an existing system or a future system. In the domain of engineering, a model is an abstraction to document a system design (Reeves, 2005). Engineers live with the models to design and communicate without considering irrelevant details in every corner of a context of use. In manufacturing industry, models describing functions and forms are used to translate needs to physical products. In IT industry, models such as UML (Unified Modeling Language)-based diagrams or source codes are used to map user requirements to compiled programs executable in run time. The mapping will exist for ever, as long as human beings cannot find a better mechanism to directly translate the reality to the finished products.



Figure 4.5 Models are used to translate reality to products.

The term “model” has different definitions in different engineering fields, but it is common that models are used to represent, to communicate, or to simulate what is designed. Models play a critical role in the scope of this study (Figure 4.6): In this scope, implementers use the system design models (e.g. UML or HCD models) to describe functions and content at different levels of abstraction to be delivered in software; production engineers use engineering design models to describe what should be manufactured. Hence, modeling activities universally translate user needs and business needs in a context of use into a product.

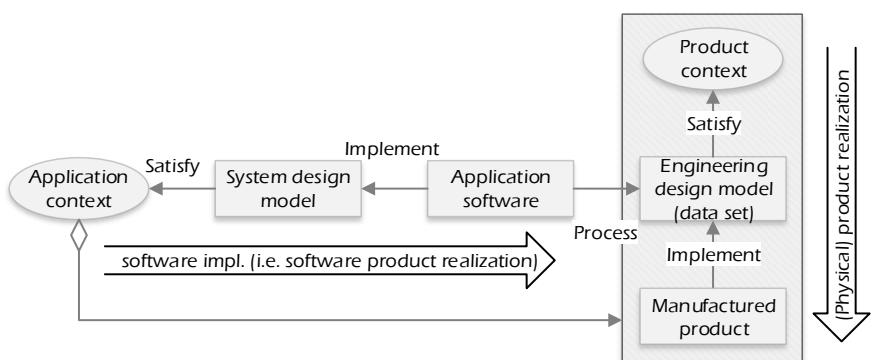


Figure 4.6 Software implementation supports (physical) product realization in manufacturing industry: The scope of this chapter.

Particularly, applications targeted on system integration need the engineering design models (the cross point in Figure 4.6) facilitating functions of interoperability: Communication, interpretation and processing. Hence, the engineering design models should instantiate standardized data schemas (upper left of Figure 4.7). This is where information model mapping is involved. The two processes, information model mapping and software implementation, are performed by implementers. Both processes share the same input and output, i.e. application contexts and application software.

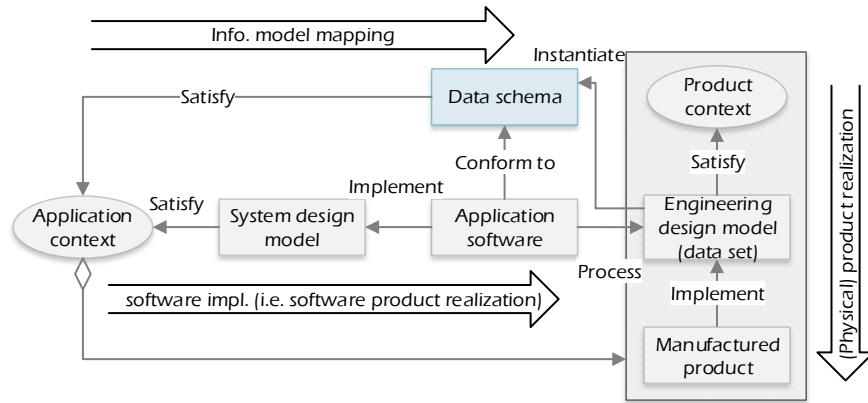


Figure 4.7 Apply data schemas for system integration.

Moreover, between the input and output, intermediate models are involved in both processes to describe what should be implemented. The intermediate models compensate the left part of Figure 4.7 for information model mapping, which becomes Figure 4.8. In this new illustration, the added activity models and reference models formulate the problems described in Section 4.1.1. Implementers should perform the two parallel processes (Figure 4.8) which are separated by shared similarities in sequence and abstraction manners. The similarities imply opportunities to save implementers' workloads by reusing the existing model artifacts during some implementation stages (Figure 4.9).

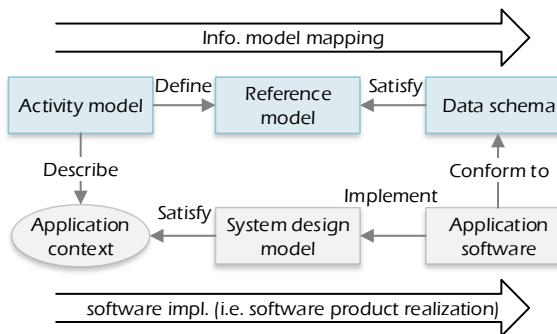


Figure 4.8 Information models are used in implementation.

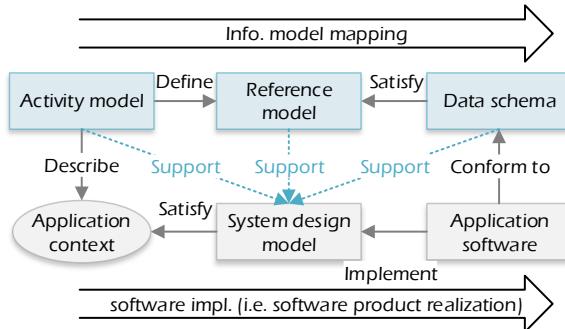


Figure 4.9 Information models have potential to support implementation.

4.2 The essence of software implementation activities

Now that the implementation need to be underpinned based on its stages, it is necessary to clarify what stages should be underpinned. Existing studies of engineering activities in general reveal basic characteristics of implementation processes performed by human engineers, in terms of actions, sequences and iterations. In the engineering context, design is a core activity and an encompassing concept in any form of implementation that performed by engineers.

It has concluded that there was no structured sequential process for engineering activities especially in a complex context (Fricke, 1996), except a common categorization of the activities being performed flexibly and iteratively: Analysis, synthesis and evaluation (McNeill, et al., 1998, Figure 4.10). This finding coincided with a triple-mode pattern that led to novel design decisions: Examination, drawing and thinking (Akin and Lin, 1995). Nevertheless, sequential definitions of the design processes in a macro scale were still recommended and generally observed. For instance, Akin and Lin (1995) made segmentation based on time stamps of protocols; they studied and concluded three segments: Problem understanding, design and retrospection, where the design segment was divided into three phases: Conception, development and representation (Figure 4.11). “Design” was used only for the second segment, and problem understanding and retrospection were defined as separated parts of the model. However, it is hard to say that designers need not understand problems before drawings and do not review their work from time to time.

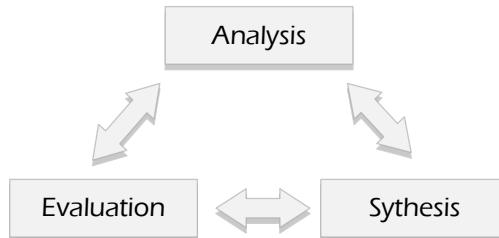


Figure 4.10 Categorization of engineering design activities (McNeill, et al., 1998).

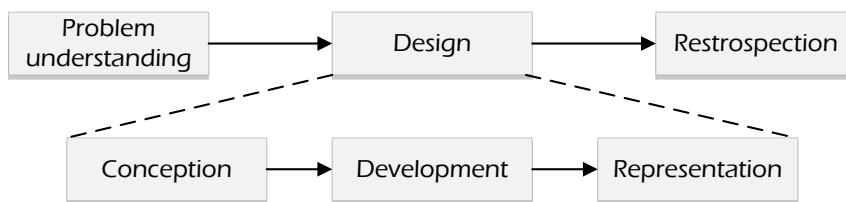


Figure 4.11 Segments of design protocols (Akin and Lin, 1995).

4.2.1 Stages of software implementation

The triple-mode pattern is universally applicable not only for engineering design activities, but also for software implementation activities. This applicability can be confirmed by widely-accepted modern software implementation styles where extensions may be made to the pattern according to specific contexts. Three styles are elementary to set the stages of all modern software implementation methodologies: 1) The waterfall is useful to understand the basic stages and the necessity to avoid iteration; 2) the incremental style and 3) the prototyping style focus on right ways of iteration.

A waterfall-styled process (Royce, 1970, Figure 4.12) was a common traditional way to describe how to ideally develop software, also based on a target of bridging the reality with produced programs (Figure 4.4). This waterfall style is useful for sequential introduction of important stages in an implementation process. The proposed process did not deny the existence of iterations, but emphasized using complete in-time documentation and stakeholder involvement to eliminate unwanted iterations. Note that,

the referenced “design” activity in Figure 4.12 was defined not as an encompassing concept, but as a limited concept to simulate and plan the coding and the testing.

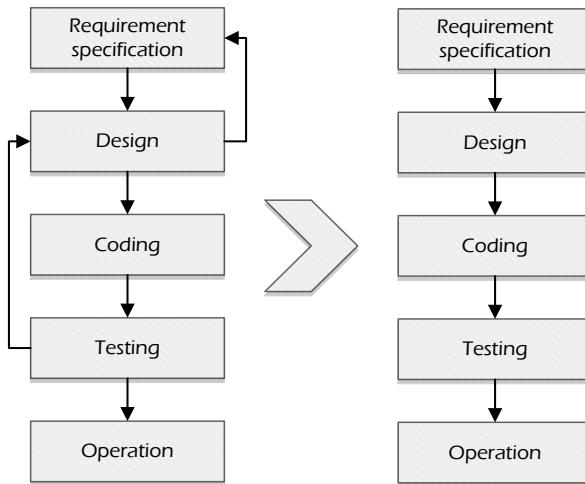


Figure 4.12 The waterfall software implementation style aimed at eliminating unwanted iterations (Royce, 1970).

However, software implementation has never been ideal. It is hardly possible to specify all functions and content, as well as complete definition of forms and behaviors of system components, at an early stage. It is always valuable to bring back the iterative fashion (Figure 4.10), even at a macro level. Hence, iterations are often inevitable and can be helpful if managed properly. There are two basic styles of iterations depending on deliveries of each iteration: Increments or prototypes. Both styles deal with limited resources and changing contexts by iterating the waterfall style to a certain extent (Figure 4.13). Reflecting analysis of results, refining products and evaluating with stakeholders, both styles practice the analysis-synthesis-evaluation pattern. These two elementary styles are the basis of almost all modern software implementation approaches.

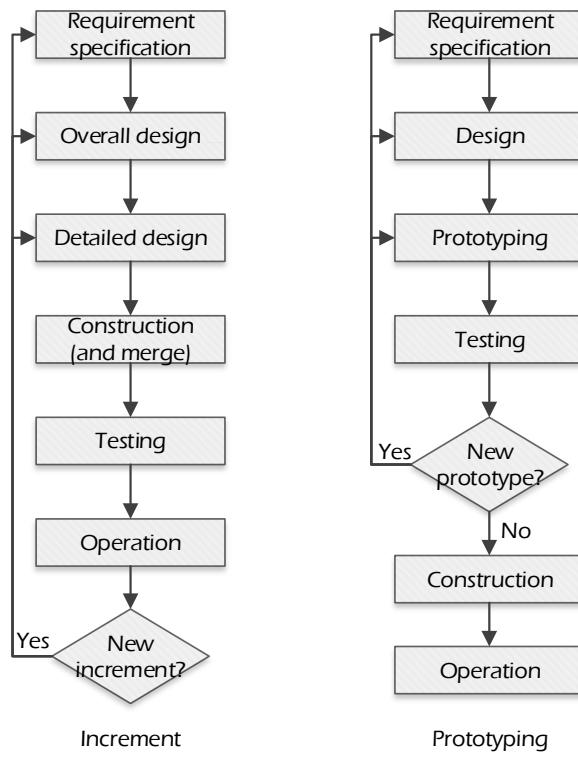


Figure 4.13 The incremental development style and the prototyping development style.

The incremental development style emphasizes design decisions at two levels: Overall design for high-level structure of all possible increments of a system and detailed design of each increment. Iteration of the incremental may begin with one of the three different phases: Requirement specification, overall design and detailed design (CMS, 2008), but the detailed design is mostly often revisited for a new increment. The delivery of each iteration should be an operational system with all developed increments merged. The incremental strategy has been widely adapted in different development methodologies. A typical example is the Unified Process (UP) with its most famous variation, the Rational Unified Process (RUP, Kruchten, 2003).

On the other hand, the prototyping style uses prototypes as a fast-developed simplified abstraction for testing to help developers refine design decisions until a design is feasible enough for formal implementation (Bally, et al., 1977). Stakeholders are recommended to be constantly involved from requirement specification to testing, especially in refining iterations of the prototypes. The prototyping strategy has also been integrated within many popular development methodologies and possibly applied together with the incremental strategy.

4.2.2 Common software implementation approaches

The two strategies with emphasis on fast and lightweight iteration set a conceptual foundation of most modern software development frameworks which compensate strategies with organizational principles at different detailing levels. At a low detailing level, *agile software development methodology* presents principles widely accepted in todays' IT industry: High adaptability, less relying on prediction, continuous delivered products, fast responding to requirement changes, more individual communication & less documentation, iteration ending with workable product, decided iteration cycle time, etc. More or less, these principles can be found in established development frameworks at a high detailing level.

Extreme programming (XP) and Scrum are two well-known frameworks to apply the agile development process at a high detailing level, which are recommended to be fully integrated in a single project. XP tends to depict practices that advocate principles of the agile development methodology and Scrum focuses on regulated organizational processes with designated roles. Both frameworks value rapid iteration but XP may have shorter cycle time (1 to 2 weeks vs. 2 weeks to 2 months). The implementation sequence of features can be adjusted in XP, but not adjustable in a "sprint" of Scrum. However, priority of features determines the implementation sequence of XP but does not affect the implementation sequence within a "backlog" of Scrum. What's more, principles of XP contribute to many other types of engineering practices as additions or adjustments of the agile development methodology e.g. continuous integration, pair programming and refactoring. Scrum does not use the practices to regulate how the framework to be applied in real work. It is possible for Scrum projects to integrate the engineering practices defined in XP.

Large software vendors also propose specific development processes that may be well suitable in some context and reflect the three-mode design activity pattern (analysis-synthesis-evaluation). Microsoft proposed a process called “synch and stabilize” when developing Windows 95 (Cusumano and Selby, 1997, Figure 4.14) to specialize the incremental strategy. The process was very suitable for a “small team” culture of Microsoft and should be effectively supported by usability labs through all the three phases. The process was iterated toward milestones defined by targeted features and targeted features were typically divided into three parts for increments according to pre-defined priorities. Nevertheless, the process served as an organizational strategy rather than a technical strategy. It did not specify modeling details to document requirements and to communicate solutions and designs. The roles of designers and programmers were blurred in the second phase because both design and coding were suggested to be performed by so-called “developers”.

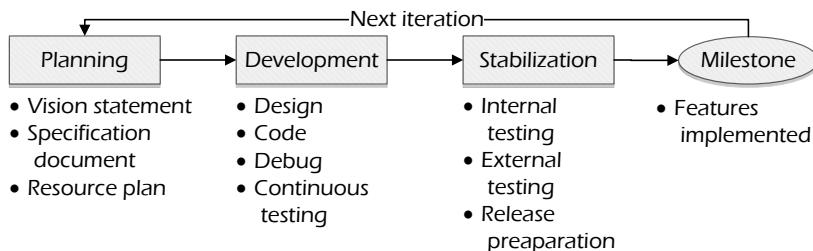


Figure 4.14 Microsoft “Synch and stabilize” process
(Cusumano and Selby, 1997).

4.3 Model based software implementation

The activities of software implementation rely on models as deliveries. System design modeling (as shown in Figure 4.6) is an important methodology to convey the ideas, reasoning and decisions of a design in software implementation. An important modeling principle is architecting based on viewpoints, driven by use cases or scenarios (ISO/IEC/IEEE, 2011b; Rozanski and Woods, 2012). For example, designated viewpoints are the basis of the 4+1 view model (Figure 4.15. Kruchten, 1995) to generate and document software architecture. Some of UML modeling techniques were selected to

develop and describe the software architectures, e.g. class diagrams, use cases, interaction diagrams and packaging diagrams. These viewpoints were supposed to feed different stakeholders of a product: Project managers, system engineers, programmers, clients, testers, etc., so that each of them could conveniently view a specific aspect of an architecture design. For instance, software implementers need the logic views to represent business logics and domain concepts based on scenarios and functional requirements; the logic view is highly similar to the information requirements (Section 3.4.3) represented in the ISO 10303 modeling architectures.

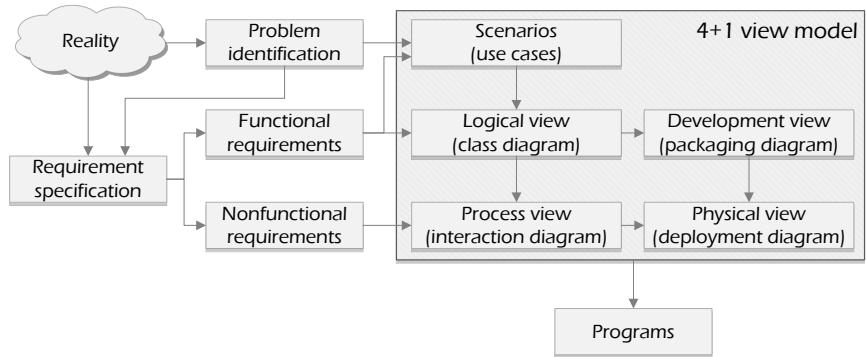


Figure 4.15 The 4+1 view model of software architecture.

The 4+1 view model demonstrated a typical viewpoint-based use-case-driven architecting process for designing software systems with probably a large amount of interaction. However, this design style has been raising criticisms for several reasons. The use cases, as a descriptive method to generate much information important for the entire development, usually left too many implicit assumptions of the system design (Constantine and Lockwood, 1999). Tasks were organized in a hierarchy, but no priority had been assigned to the tasks. There was also a lack of description of systems' precise behaviors, which resulted in low efficiency and low flexibility during the development process (Cooper et al., 2014).

4.3.1 Involvement of the HCD paradigm

Such criticisms lead to new ways of architecting software concentrated on effectively modeling user needs in specific context of use, i.e. the HCD (Human-Centred Design) paradigm. In addition, the HCD paradigm meets the

needs of the CAx systems characterized with intensive human-computer interaction in the engineering context. Development of such systems demands effective information architectures for rich content presentation and manipulation.

Many HCD methodologies adopt the stages set by the waterfall style as an initial start point, with some extensions and iteration points. Sharps, et al. (2007) exemplified this style with a typical four-stage process (Figure 4.16). The setting of the sequence could be more flexible, which was argued by the influential “star model” (Hix and Hartson, 1993, Figure 4.17). It revealed high flexibility of the design process where designers should be free to connect the activities in any sequence if only through evaluation, rather than stuck to sequential or iterative rules. On the other hand, its strength also became its weakness in real use, because it made a project unmanageable, where no specific hint about progress was provided. Perhaps this was the reason why people reached a consensus of specifically defined interdependences of HCD (Human-Centered Design) activities (ISO, 2010b, Figure 4.18). In this standardized HCD process, the activity of evaluation was still a major trigger of iteration, but other activities became sequentially interdependent.

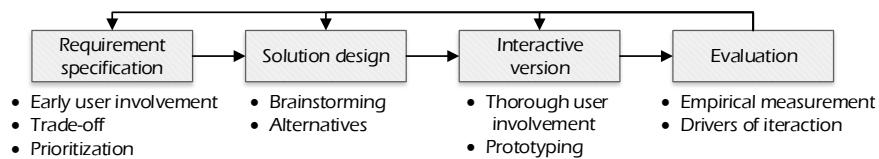


Figure 4.16 Four activities of interaction design (Sharps, et al., 2007).

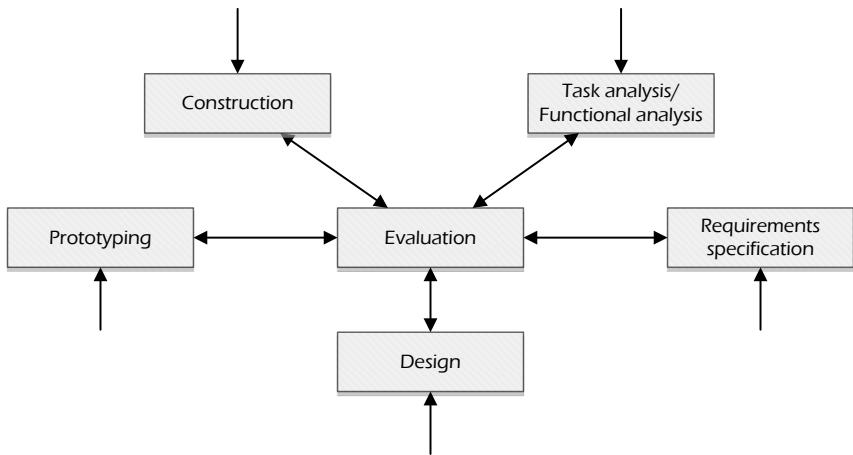


Figure 4.17 The star model (Hix and Hartson, 1993).

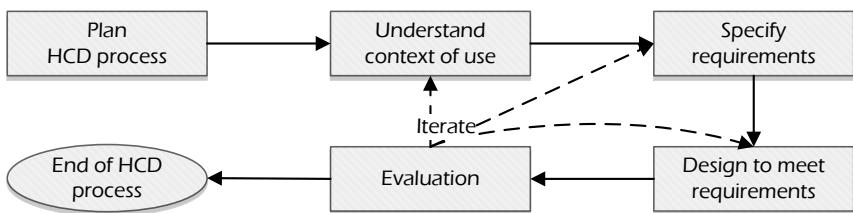


Figure 4.18 The standardized HCD process (ISO, 2010b).

Note that the mentioned processes mainly focused on the design activities isolated from a bigger picture, i.e. the technical process of application software lifecycle (ISO/IEC, 2008). To accomplish a final product delivering desired user experiences, cross-disciplinary effort has to be required. Thereby, Mayhew (1999) made an exceptional summary to integrate the HCD process with software engineering activities, specifically fitting the designers' activities into the process of OOSE (Objective-Oriented Software Engineering, Jacobson, 1992). The result was called usability engineering lifecycle (Figure 4.19) which focused on modeling of every contextual element (e.g. users, domains, tasks and interactions) before detailed UI design.

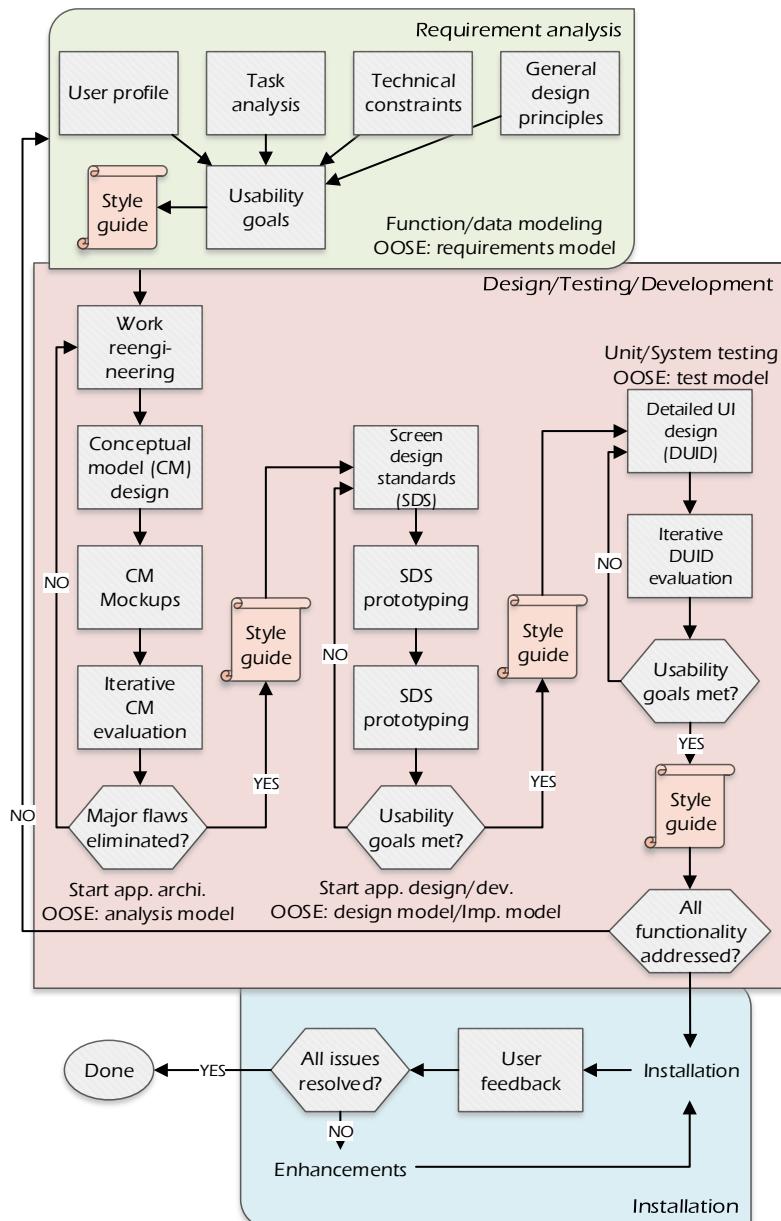


Figure 4.19 Usability engineering lifecycle (Mayhew, 1999).

To sum up, models have been used in many approaches for software implementation, especially when integrating the HCD principles. There are pros and cons about the HCD. The HCD aims at avoiding too many iterations by accurate specification of user requirements and comprehensive understanding of content. That said, the HCD-based concerns are limited when it comes to either software architecting or software engineering, which may draw little attention of implementers. Namely, integrating the HCD principles exhibits more contextual modeling but less focus on technical design modeling. A generalized process can be derived based on studies on previous important implementation processes (Figure 4.20) for which the iterative fashion is encouraged with evaluation in each step.

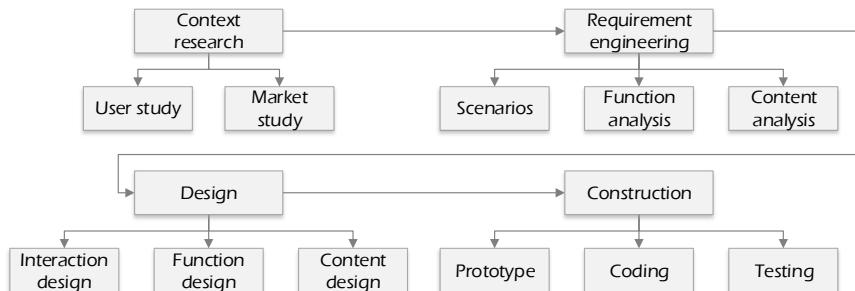


Figure 4.20 A generalized process of software implementation.

4.3.2 In-depth participation of domain experts

What is more, the CAx systems are often too content-rich to achieve best UX (user experience) only by practicing design principles in software engineering and HCD. When the rich-content domain knowledge forms the bottleneck of the implementation, it is relatively easy for domain experts to dominate development of system architectures. Therefore, involvement of end users (i.e. domain experts) are highly recommended to ensure professional integration of domain knowledge. Based on observations, many system integration projects in industry to solve interoperability issues have been (partly, at least) driven by primary users (end-users or operators) of the systems. The implementation cases in the appended paper B, C and D are all developed closely involving stakeholders from the user side. This fact should be taken advantage of in an integrated implementation process.

Therefore, the result of this study is prepared mainly for the domain experts with little experiences or knowledge in either software engineering or information models. It enables delivering feasible design solutions ready for professional implementation (demonstrated in Section 5.2 and 5.3). Hence, this study focuses on generate architectural decisions related to domain knowledge. In general, deliveries of different activities in an implementation process are essentially sets of decisions with descriptions at different levels of abstraction. In this study, the implementation activities are recommended to be executed at a low detailing level, i.e. an architectural level (more details on system architectures in section 2.4). Thus, practitioners can agilely focus on decision-making to satisfy high-level implementation goals. Moreover, iterations with prototyping or increments can be analyzed, synthesized and evaluated in an agile fashion.

4.4 Process integration

The design principles and focuses are different between the software engineering perspective and the HCD perspective, which forms a gap between the software engineering methodologies and the HCD methodologies. There is another inevitable gap created when implementing with the information models, where implementers could expect heavy workloads (Section 4.1.1).

As discussed in Section 4.1.2 and 4.1.3, the potential of information models (i.e. rich semantics and procedural similarity) makes the information models useful for the entire software implementation process. Thereby, when implementing with information models, it could be valuable and promising to investigate the fitness of integrating the model mapping process with the established workflows of software engineering and HCD (Figure 4.21).

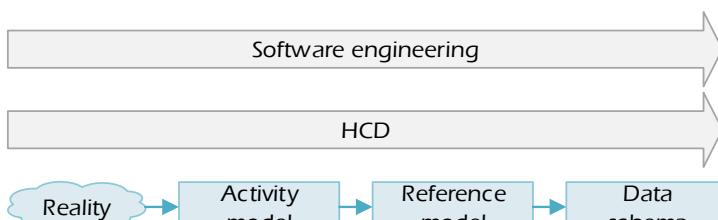


Figure 4.21 Processes need to be integrated.

The intermediate models (activity models and reference models) provides different levels of abstraction that is practically useful for application design. There are usually a hierarchy of multiple artifacts at different levels of abstraction to construct an activity model or a reference model. Hence, specific parts of these models can be related with specific implementation stages, according to the levels of abstraction. This relationship can be illustrated in Figure 4.22. This illustration shows that the existing artifacts of modeling architectures, as references, support different stages of the general stages of the conventional software implementation workflow. Note that this illustration implies the order of timing and abstraction levels only between adjacent entities linked by solid arrows.

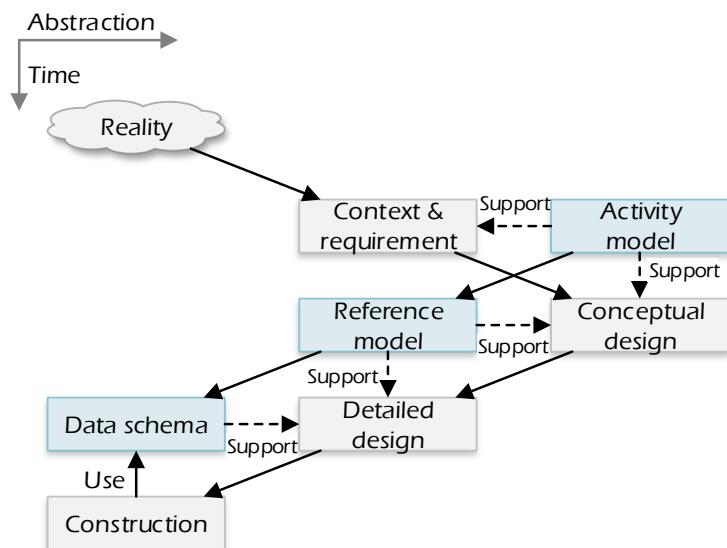


Figure 4.22 The information model mapping process underpin a process to implement software at different levels of abstraction.

This supportive relationship in accordance with sequence and abstraction levels is the basis of integration of the different processes. The following Figure 4.23 is a brief illustration of this process integration in two dimensions. Horizontally, the general software development process is extracted from Figure 4.20. This process, to a certain extent, fit with both the

software engineering process and the IxD process which pay different levels of attention in different stages. Vertically, the information model mapping process is deployed for suitable stages of the implementation. Integration of the two dimensions creates a model driven software implementation process for system integration.

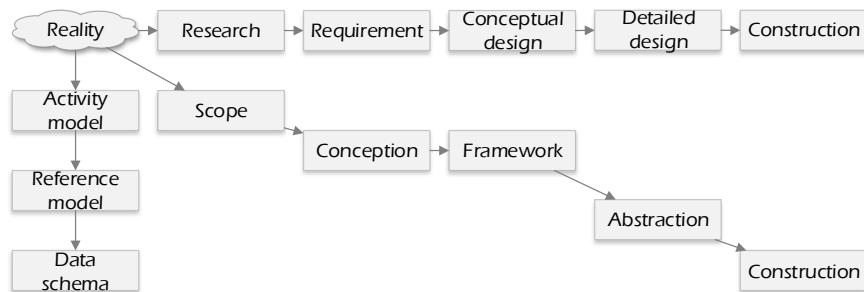


Figure 4.23 Integration of the processes.

Perhaps it is too conceptual to understand the process only from a two-dimension view, as there should be more detailed reasoning on how this integration actually works. This process can be detailed and illustrated as in the following Figure 4.24. Each stage is dependent on the previous one and there are two or three key activities for each stage. As recommended by trendy methodology of agile development (Brhel, et al., 2015), an iterative and incremental fashion based on timely evaluation should be applied for this lifecycle, rather than the implied waterfall style. No decision should be set in stone for each stage, otherwise the project may suffer from low flexibility in a changeable application context.

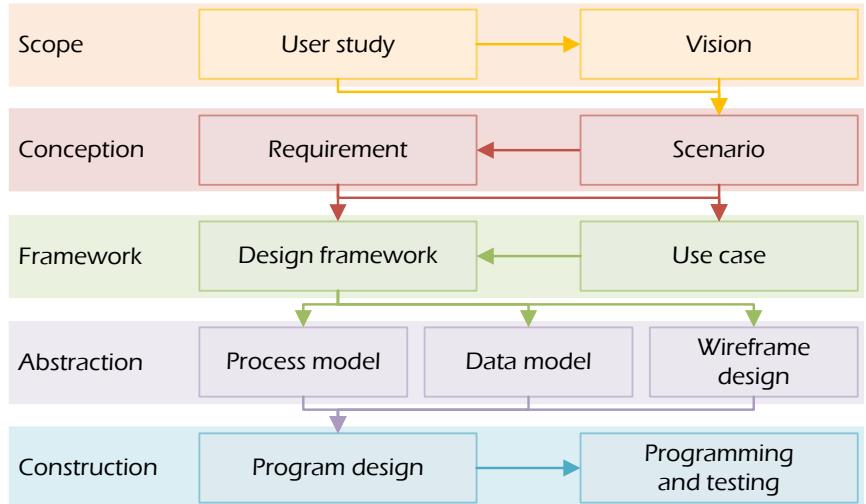


Figure 4.24 Stages and activities of the implementation process.

As mentioned in Section 4.3.2, this study is mostly concerned with interests of domain experts as key implementers for system integration. Although the process can be traced down to a construction stage, this thesis focuses on delivering architectural design decisions. That is, deliveries in most activities are descriptive artifacts at a low detailing level, i.e. contextual study for scoping, conception based on a narrative description, concrete design framework, a selected set of UML-based descriptive models and a set of lightweight wireframes. Therefore, even industrial practitioners or end-user developers with little experiences in software implementation are able to perform this process and deliver feasible solutions ready for professional construction.

In terms of sequence and abstraction levels, the integration at the level of activities can be illustrated in Figure 4.25 together with the traditional processes of HCD and software engineering. Most of the initial activities to analyze contexts and requirements can be supported by interpretation of the information models so that the effort of the model mapping is not only liable to the implementation. This integration can solve a key problem of the modern agile fashion to develop software, i.e. a lack of reliable domain-specific models in early stages or early iterations. The information models can be a critical support as a domain knowledge base for the developed applications to guarantee comprehensive data models and process models.

The integration is also a bridge for the separated activity concepts in the HCD and the software engineering. The HCD process is expected to be practiced in any type of development processes, but software engineers are used to focus on rapidly delivering viable products instead of acceptable user experiences. After all, not all systems need to be interactive. However, most system integration in the manufacturing industry is all about efficient interaction with engineers in the subject domains and HCD principles will be highly useful to deliver valuable applications. The rest part of this chapter will introduce each stage of this process with examples and references to appended publications.

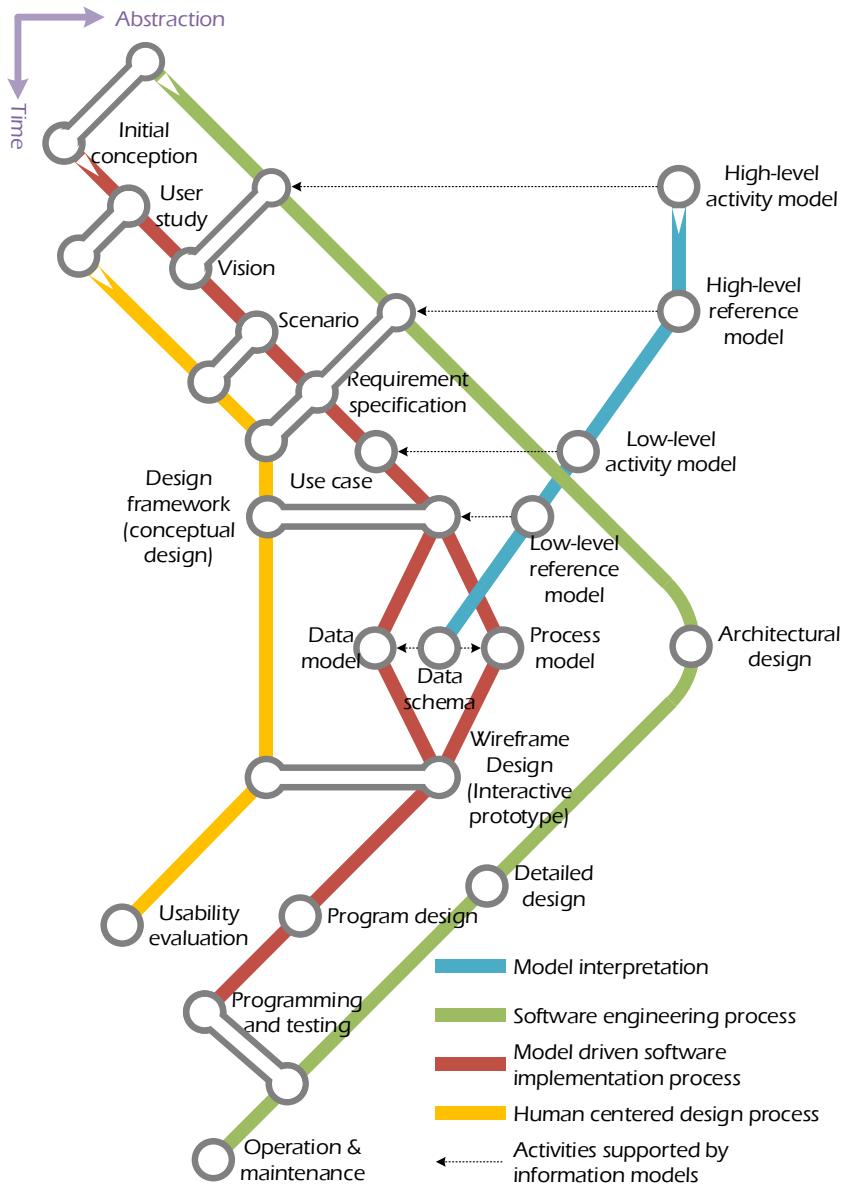


Figure 4.25 Activities of *Model driven software implementation process for system integration* and the parallel processes.

4.5 The scope stage

At first, scope of the application should be defined, which is also a key function of activity models describing processes from the perspective of the subject domains. Activity models provide first-hand informative support for goal identification and most of behavior patterns to study users. For instance, the ISO 14649 STEP-NC (ISO, 2003b) provides activity models generally describing (not prescribing) typical understanding of the process from design to manufacturing of products represented in IDEF-o diagrams. The highest level of these diagrams is shown in Figure 4.26 which can be decomposed into lower levels of diagrams. The diagrams display domain specific information on data flows, functionalities and contexts.

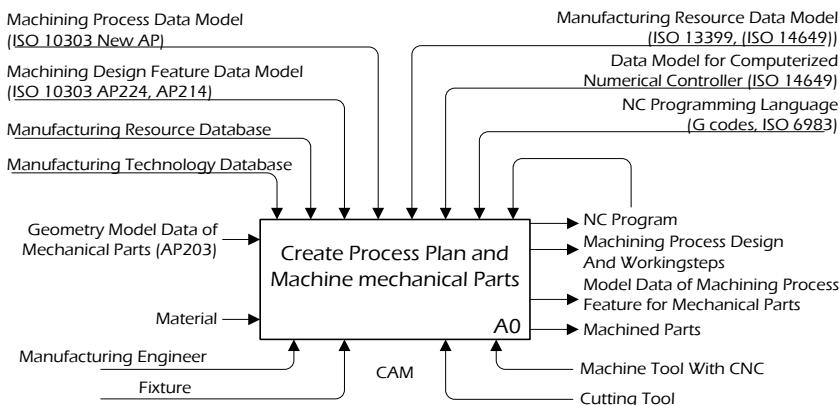


Figure 4.26 The highest level of the activity models defined in ISO 14649.

Designers are supposed to identify and select a part of the activity models that are useful for the targeted direct users of developed applications and to define the scope, based on the activity models in higher levels and other available resources, e.g. user research, market research and literatures. The appended paper F is an example of an extensive literature review on the scope and the context of engineering applications in manufacturing industry. Mature standards such as STEP-NC are often developed by industrial practitioners who are familiar with the application contexts. Therefore, the information models at this level can be greatly useful when there is a lack of resources to perform a comprehensive contextual study.

4.5.1 User studies

Software engineers traditionally consider their work starting with requirement analysis, but this convention is criticized by experts in IxD (Interaction Design), IA (Information Architecture) and UX (User Experience). Research on users' needs and goals is a preferable initiation for most interactive system design. Modeling users to describe personal information of the target users is useful when system objectives are unclear before defining any requirements.

Persona (Figure 4.27 is a simplified example) is one the most common time-tested way to study and model the users. It is a synthesized depiction of targeted typical real users to study goals, behaviors and attitudes of the users. Abstraction and composite of these factors formulate the personas which assist the decision, design and evaluation of products.

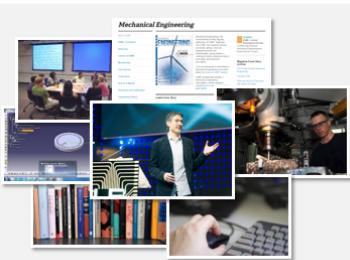
USER STUDY · A PROTO-PERSONA Brandon  <p>"People forget how fast you did a job – but they remember how well you did it"</p> <p>AT A GLANCE AGE – 40 LOCATION – Stockholm JOB – Senior process planner</p>	BEHAVIORS ANALYZING PROCESS PLAN INFO – Understanding process plan info associated with product design info, and identifying potential risks. SYNTHESIZING RISK MODELS – Establishing risk interdependency, locating critical potential problems, and addressing countermeasures EVALUATING RESULTS & UPDATES – Being aware of changes in all models, influences of the changes, and completion states of the countermeasures; and adjusting the risk models EXERIENCE NEEDS ATTENTION - Staying focus on most critical issues CERTAINTY - Feeling reassurance about all the issues REUSE - Avoiding "reinventing the wheels" INTEGRATION - Info in an integrated, controllable, tangible manner FLEXIBILITY - allowing a non-sequential, back-and-forth freely, iterative process LIFE MOTIVATORS BE PRODUCTIVE – Brandon is proud of his productivity and punctuality as is also part of organizational culture FEELING ORGANIZED – Quality assurance is a critical work, but not his major work. He needs to switch easily in-and-out with this temporary team LEARNING MORE – Not just for expertise in process planning, he'd like to learn more about quality assurance, IT tools, product design, and line design
--	---

Figure 4.27 An example of proto-persona for a user of risk assessment applications.

It is often unlikely to perform exhaustive classic user research obtaining detailed qualitative data for urgent CAx-related implementation in industrial reality. Hence, proto-personas as a provisional version of formal

personas are recommended as a tangible user model that less resources are required (Buley, L. 2013). A proto-persona usually explains users in a less narrative and more structured way than traditional personas. A proto-persona (exemplified in Figure 4.27) as a tool is not an accurate method to generate and validate a design, but it helps designers focus their priorities on critical needs, critical tasks and critical workflows. Also, it provides a concerted empathetic view for team discussion during the implementation process.

For a user study, goals are drivers of behavior patterns that model the users. Individual goals are actually the ultimate concern of UX studies (Bevan, et al., 2015). In the context of personas, modeling the users is essentially descriptions of goals at different levels. It is always important to remember that the design of interactive systems should be goal-directed, where goals (why users act) motivate behaviors (how users act). After all, it is possible that users may act ineffectively or inefficiently in their daily lives, but their goals are correct. Hence, the user models should elicit the true motivations at first. Researchers may not be able to identify the user goals immediately. Sometimes, designers can understand the application domain so well that goals can be directly identified, but in most of cases it is a concentrated study on users that is needed to generate a comprehensive view of user goals.

Norman (2005) proposed three levels of cognitive and emotional processing: Visceral, behavioral and reflective. They can be mapped to three levels of user goals: Experience goals, end goals, & life goals (Cooper et al., 2014):

- Experience goals define how users want to feel: Comfortable, pleased and enjoyed, although it is difficult to talk about the experiences “in the context of impersonal business”.
- End goals motivate performing tasks. Users’ “behaviors, implicit assumptions, and mental models” need to be complemented by product behaviors or product functions through tasks.
- Life goals represent personal aspiration beyond the subject context and explain why end goals need to be accomplished.

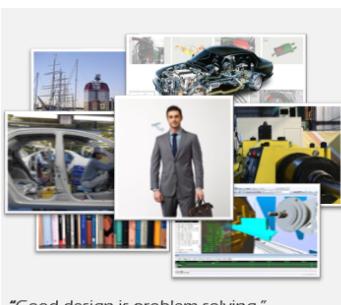
End goals are considered as a core part that is concerned mostly by designers, especially in the context of business application development, so in proto-persona the end goals with related behaviors are addressed at

first. The three levels of goals are titled in an easier-to-understand way, i.e. behaviors, experience needs and life motivators.

In the appended publications for kinematic data exchange (paper A and paper E), the needs are derived directly from a local automotive producer. In the beginning, system scope looks confusing until the users' needs are clarified (see Figure 4.28). Users for cutting tool data exchange, with classification, GD&T (Geometric Dimensioning and Tolerancing) and catalogues (paper B and paper C), share a similar model with kinematic data exchange. They are both expertise on process planning and manufacturing resource (e.g. machine tools and cutting tools) management and both play a similar role for their daily tasks.

USER STUDY · A PROTO-PERSONA

Scott



"Good design is problem solving."

AT A GLANCE

AGE – 30

LOCATION – Göteborg

JOB – Process planner

BEHAVIORS

ANALYZING AVAILABLE INFO – Understanding product design info, with other info regarding manufacturing resources such as machining tools and factory layout design.

SYNTHESIZING MODELS – Proposing process plans synthesized with available information

EVALUATING PROCESS PLANS – Checking the process plans along with production line models.

EXPERIENCE NEEDS

CLARITY – Feeling clear about all the information

CERTAINTY - Feeling reassurance about all issues and no missing information

REUSE - Avoiding "reinventing the wheels"

INTEGRATION - Info in an integrated, controllable, tangible manner

LIFE MOTIVATORS

BE PRODUCTIVE – Scott tries to keep his productivity as required by the fast-changing industry.

BE RESPONSIBLE – Although quality assurance is not his major task, he'd like to check his design as throughout as possible before handing to others.

Figure 4.28 The user model for users of kinematic data exchange.

4.5.2 Visions

User models describe the motivators and behaviors from the standpoint of users. Now it's time to address a vision from the user models to satisfy user goals at a high level. Goals and, perhaps, frustrations have been highlighted, so that consensus about priorities can be established. Defining a vision essentially includes two steps both in a high level, i.e. a problem statement and a vision statement. The problem statement focuses on current frustra-

tion with a situation that needs to be changed and a cause-and-effect relationship between the frustration and the user goals. The vision statement addresses design objectives with a feasible answer to that problem by the provided product. Take the development of a system for risk assessment of process plans for example:

PROBLEM STATEMENT

Currently PFMEA (Process Failure Mode and Effects Analysis)-based risk analysis is performed in an unsatisfactory manner. There is a lack of systems to support necessary tasks to guarantee accurate and efficient quality assurance. The tasks include information integration, contextual analysis, problem identification, solution proposal, activity tracking and update management.

VISION STATEMENT

A model driven system design for PFMEA-based risk analysis will help users achieve accurate and efficient quality assurance. Highly integrated models are demanded to facilitate a holistic GPS (Geometrical Product Specifications)-facilitated view and a flexible interaction manner. Issues regarding fragmented data, untraceable workflows and stubborn spreadsheet-based interaction should be avoided.

The key part regarding information model is the information integration in the problem statement and the fragmented data in the vision statement, both of which reflect the large number of information sources for process planning displayed in the high-level activity model (Figure 4.26). Thus, prioritized expectations can be stated easily based on the problem and the vision. In the case of risk assessment, the prioritized expectation can be stated as: 1) A holistic view based on 2) a fully integrated model with all pieces of information (shapes, properties, features, risks, resources, & processes) becoming manageable (retrievable, in a context, interrelated, reasonable and versionable) supports 3) a flexible workflow and 4) a traceable result (see Figure 4.29).

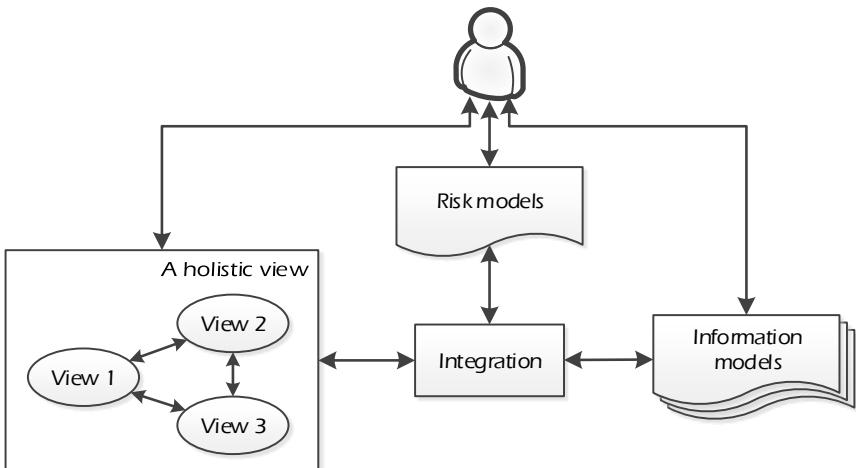


Figure 4.29 A vision proposal for a risk assessment system.

Sometimes a feasible solution is not easy to generate due to limitations in technology and business logic. For example, representation and exchange of digital product catalogue have been a critical issue in the engineering context (paper C). In an industrial sector, such as cutting tools, vendors, users and system suppliers often use different views and representation methods of information, which makes communication between the parties difficult. The interoperability issue can be solved by system integration with three functional units each of which interprets and presents different types of data (Figure 4.30). The units can be implemented as independent applications or integrated as plug-ins with existing systems.

For system integration, it is possible that no requirement on functions but the interoperability as a quality requirement needs to be implemented in an existing system. The kinematic data exchange for CAD / CAM systems exemplifies this scenario (paper E). Siemens NX and STEP-NC Machine are two systems possessing different exchange structure of a same type of information, the kinematic mechanism. Two plug-ins should be implemented to blend in the existing systems and existing workflows (Figure 4.31). The two plug-ins are named as KIBOS for NX and KIBOS for ST-Machine.

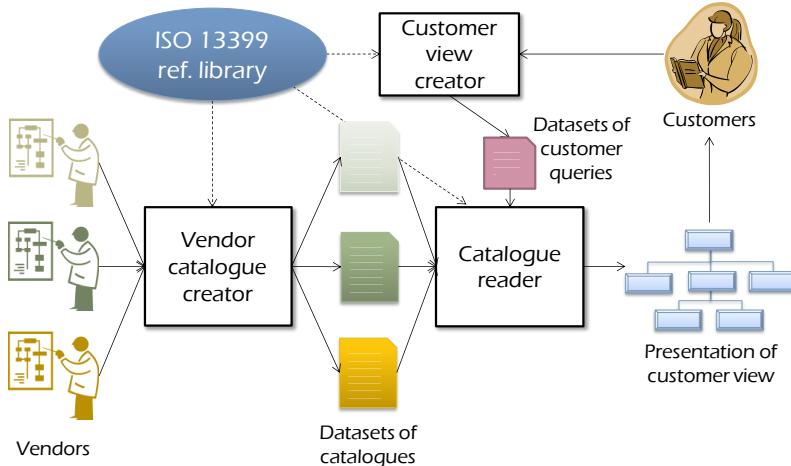
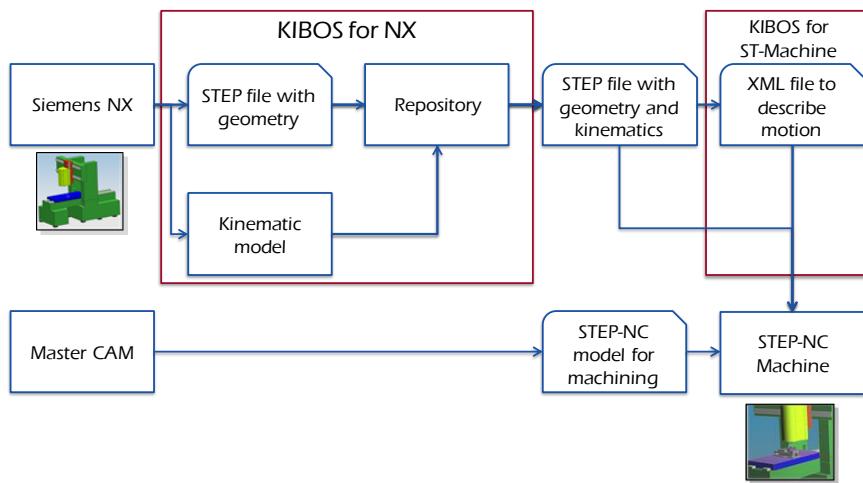


Figure 4.30 A vision proposal for a digital catalogue exchange system.

Figure 4.31 A vision proposal for kinematic data exchange.
("KIBOS" denotes KTH Implementation Based On STEP)

4.6 The conception stage

In this stage, a problem space of the context should be understood at a more detailing level so that designers identify *what* kind of applications can meet users' needs before defining *how* the applications should behave.

Moreover, traceability should be established to justify *why* requirements are stated (Rubin and Goldberg, 1992). Technically, the previous step has already done a part of the job, because the user goals are often considered as an important part of scenarios (and even use cases). In this stage, the user models are fitted into a more extensive context (Kuutti, 1995), the scenarios. Also, the designers can extrapolate the requirements of functions and content based on the scenarios in a narrative form.

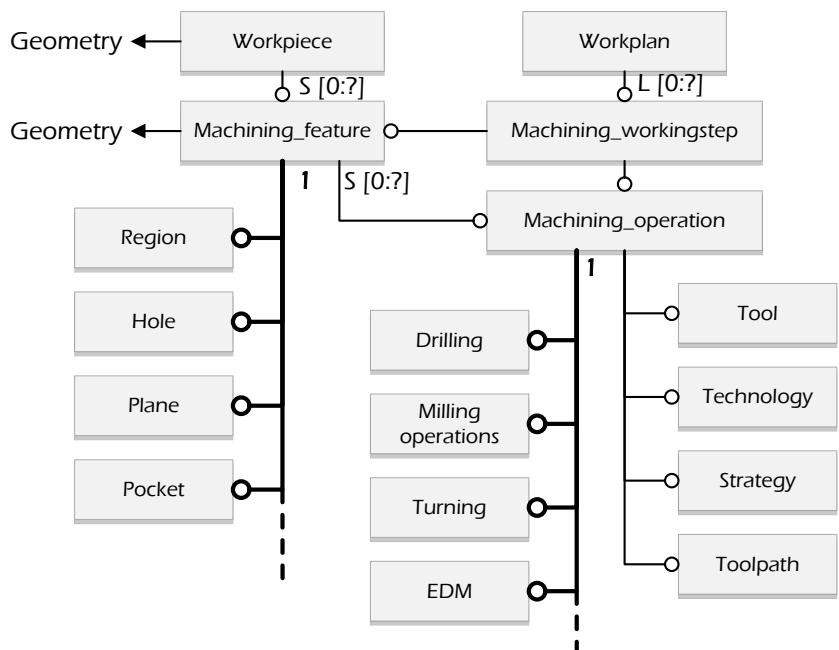


Figure 4.32 A high-level reference model specified in ISO 14649 (author reproduction based on ISO, 2003b).

Usually brainstorming is recommended in this stage because there may be too many inputs and ideas on the carpet. The high-level information models, e.g. the ones in the hierarchical activity models and the reference models, can significantly reduce the chaos. Designers can define the requirements on functions and content based on explicit illustration of activity models at lower levels. The reference models with structured and grouped metadata are essentially a formalized example of IA (Information Architecture) of the application context. Figure 4.32 displays a reference

model at an overview level for information models of computerized numerical controllers (ISO, 2003b). This kind of semantic information is important to form content requirements and functional requirements. The designers may abstract the information models and may incorporate the abstraction with other types of resources to establish a foundation for integration architectures and information architectures.

4.6.1 Scenarios

Software architecture design is recommended to be triggered by scenarios which are brief narrative descriptions for expected use of the new system by the personas with specific goals (i.e. the application context). Normally the scenarios should present the following information (Cooper, et al., 2014):

- In what setting;
- Amount of time;
- Frequency and interruption;
- Teamwork;
- With what other techniques;
- Primary activities leading to goals;
- Expected end results;
- How complex for users with specific skills and use frequency.

Context scenarios briefly imply users expected functions, priority of functions, interface styles and interaction styles. Details about product functions and interactions are not revealed, nor are implementation details. Existing products and technologies should not limit the context scenarios (Cooper, et al., 2014), unless some information is an initial constraint (e.g. for quality assurance activities in some industry sectors, PFMEA is a standard that users shall conform to). For instance, below are context scenarios of risk assessment process based on PFMEA:

1. A temporary team is formed for risk analysis to exploit senior expertise from multiple departments, with a moderator nominated. Latest digital models and previous risk analysis data are prepared, integrated and presented by the system. RPN (Risk

Priority Number) Criteria are set. The interface is usually displayed to the team via some digital hardware (possibly remotely).

2. Quality issues are assessed within the scope of a process plan in a flexible iterative fashion. The assessment may begin with a view of a feature, an operation, an executable step, a setup or a type of resource, although risks will be identified for a process eventually.
3. For each risk being analyzed and specified, the moderator will quickly locate related PPR (Product-Process-Resource) information for the discussion and assign properties and indicators for the risks. The team may need suggestions about naming, description and indicators according to previous risk analysis models.
4. Critical risks in terms of failures are highlighted, with a diagram analysis generated in a real-time fashion. Countermeasures for the risks can be specified in real-time, or in the end of the discussion.
5. Completion of countermeasures or changes in the related information models may trigger revision of the risk analysis. The influences of the changes on PPR models are highlighted, e.g. changes in product design as causes or effects may eliminate, produce, or exacerbate the risks.
6. The results will be exported and archived as an interoperable data set which should be computer-interpretable for control plans and work instructions.

Sometimes, particularly common for system integration, the scenarios are descriptions of enhanced existing workflow. The goal is to blend the developed applications into existing systems to facilitate the interoperability. The context scenarios for kinematic data exchange is one of such cases:

1. In a CAx system, a product design model with geometric and kinematic information is prepared, of which a kinematic chain is completed in an engineering design activity.
2. A function to export/save the design model with both types of systems in a standardized AP242 model is invoked and a name and a location of the exported data set are specified.
3. In another CAx system, the exported AP242 data set is selected and opened/imported to the working area, where kinematic analysis or simulation can be performed on the model which also integrates geometric information.

4.6.2 Requirements

Generally speaking, a requirement is an exhibitable property to practically solve problems (Abran, et al. 2004). In the domain of software engineering, it can be defined as “a need and its associated constraints and conditions.” (ISO/IEC/IEEE, 2011a, more details about application requirements and information requirements in Section 2.2.1) To develop a complex system such as enterprise system integration, requirements are relatively independent of product definition (Garrett, 2010). It is the requirements that define “what” the product should be before jumping to decisions about “how” it should be implemented. System designers have to, in an early stage, understand users’ objectives, business needs, success metrics, state of the art, technical requirements and priorities. This fashion provides flexibility for designers to explore all possibilities beyond technological constraints. Clear requirement definition also addresses validation criteria to evaluate fitness of a design (Cooper, et al., 2014). It avoids an illusion that performance of products can be increased by simply piling up functions and features.

Most requirements can be directly extracted from the scenarios and an ideal format may consist of objects, actions and contexts (Shneiderman and Plaisant, 2004). In this way, the requirements on functions, data and, to an extent, performance can be expressed. Clarity and accuracy are most important for definition of requirements because its high influence of following design activities. Hence, requirement engineers proposed basic

principles to define SMART requirements (Mannion and Keepence, 1995) borrowing from the domain of management psychology (Doran, 1981):

- Specific,
- Measurable,
- Achievable,
- Relevant,
- Traceable.

Based on the context scenario example for risk assessment provided in the previous section, the requirements can be elicited following the same sequence as defined in the scenarios (note that this is a refined version after one prototyping iteration):

- The system records information about the team members, the moderator, time and place during initialization.
- Relevant data for risk assessment is gathered and integrated from related information models or legacy documents, i.e. previous risk assessment results, process plans, product design and production line design, during initialization.
- Criteria of highlighting all indicators of failures (severity, occurrence, detectability and RPN, IEC, 2006) are clarified and recorded, during initialization.
- The system simplifies the above steps by reusing previous settings if the session is triggered by updates.
- Relevant data elements (working steps, operations, risks and features) based on standards on information model of process planning (ISO, 2003b) and PFMEA (IEC, 2006) are integrated and displayed textually and graphically.
- Any relevant data element can be navigated, identified and selected freely from an interface.
- Related PPR information is provided for discussion when a risk is identified.
- Information in terms of modes, effects, causes, detectability and countermeasures are input by the moderator when specifying a risk.

- Suggestions regarding naming, description, reasoning and indicators are provided based on previous risk assessment results when specifying a risk.
- Whenever a risk item is added or edited, critical risk items are updated and highlighted in a real-time fashion and so is an automatically synthesized report for criticality with diagrams.
- The results will be exported and archived in an interpretable way for authors and readers of control plans and work instructions.
- Updates of relevant data elements and related models of process planning, product design and manufacturing resources are highlighted and summarized if the session is triggered by the updates.
- A revision report will be exported if the session is triggered by the updates.

It can be observed that the application context may result in compromises on whether to specify implementation details. Normally, the technological concern should be addressed as little as possible. However, terms regarding specific technologies may be mentioned in the requirement list, e.g. information models and standardized terminology which have been specified as constraints by certain user groups.

A flow chart with swim lanes (exemplified in Figure 4.33) can be served to display how these requirements are connected, which represents users' needs sequentially. The textual expressions are simplified into phrases in blocks and lines with arrows indicating the sequence of how users accomplish their goals. Although there is a direction for each arrow implying a main path, in most cases of design activities, systems should allow users to go through the required functions back and forth, rather than to follow any specified sequence. The swim lanes provide categorization of the requirements, where the categories indicate a generalized three-stage cycle (analysis, synthesis and evaluation) of any design activities (McNeill, et al., 1998, more details in Section 4.2). The requirements essentially should support all stages in this cycle. Nevertheless, the system integration in this thesis does not always need to support this cycle, when applications are built as an enabling system only for increasing interoperability. This will be exemplified in the following case for kinematic data exchange.

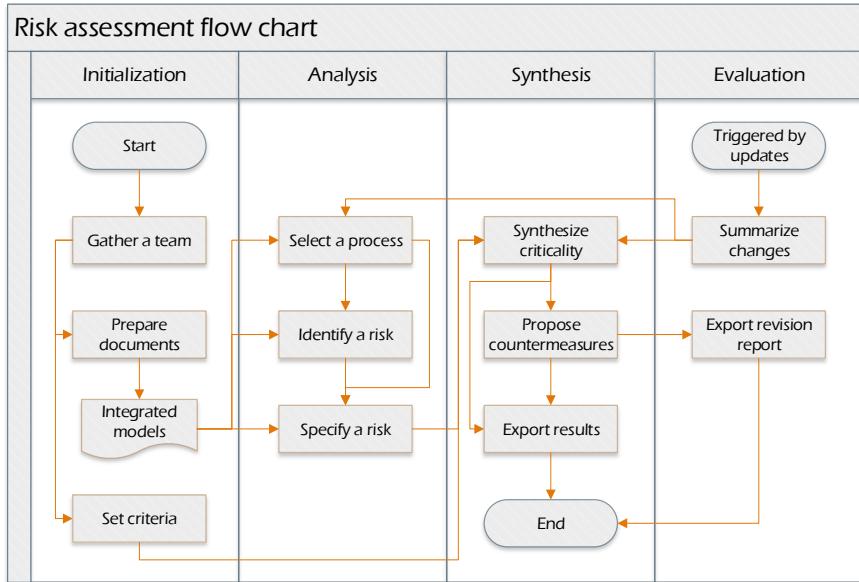


Figure 4.33 Requirements of risk assessment application.

Another example is for kinematic data exchange, where the scenarios are simpler than the risk assessment system. The interoperability as a quality requirement is more concerned, in this case, than any functional requirement. Moreover, no additional function, except basic I/O operations, should be implemented so that the application can easily blend in. This is a typical application with simply functionality but potentially complex data structure. Based on activities models and reference models in the relative information model standards (ISO, 2014a; ISO, 2014b), relevant knowledge about terminology widely accepted in industry can be easily obtained. Here is a list of requirements explored from the context scenarios:

- Completeness and validity of kinematic model is confirmed.
- Special kinematic links (e.g. the workpiece and the cutting tool for a machine tool) are identified.
- The export/save function is chosen by the user.
- The name, the location and the type of the exported/saved data set is specified by the user.
- The data set is exported as configured.

- The import/open function is chosen by the user.
- The targeted data set is located and selected by the user.
- The selected data set is imported.
- The selected model is integrated with other functional units and displayed in a proper module of the existing systems, e.g. in a machine tool library or in a working area.

Again, the following flow chart (Figure 4.34) illustrates how these requirements are related. The three swim lanes show three parts of the development that can be separately implemented as standalone applications or plugins integrated with corresponding existing systems. The three beginning points indicate the three parts are not necessarily executed one by one in a real setting, although together they achieve the interoperability between some designated systems as the implementation goal.

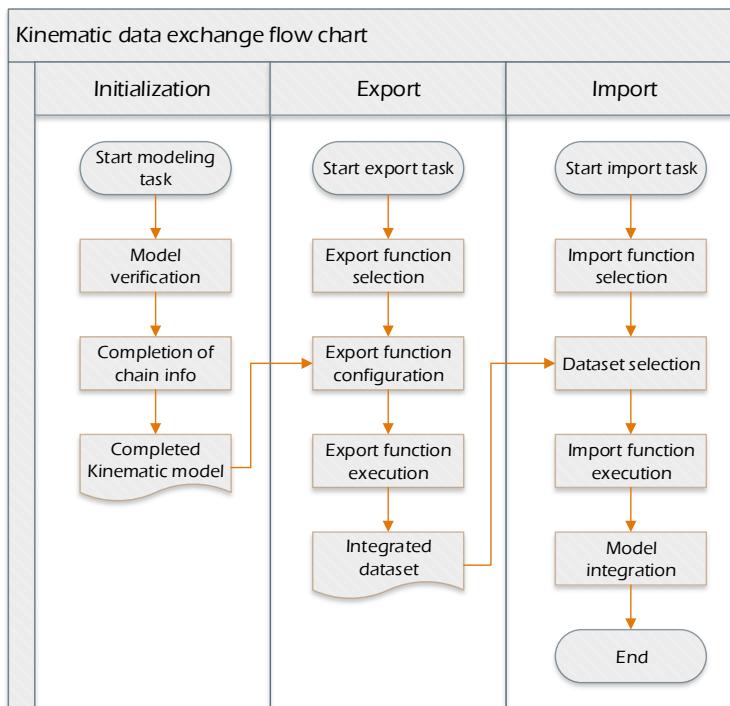


Figure 4.34 Requirements of kinematic data exchange.

4.7 The framework stage

The next stage, framework, is where design starts. Scenarios and requirements are translated to a framework which specifies the overall structure of the application software. From the perspective of software engineering, this stage can be seen as a high-level architectural design phase (see more about software architecture in Section 2.4). In this stage, detailed use cases, general description of system components and data models are generated. Use cases in a concrete fashion are greatly useful to generate, refine and validate the framework. It is also important in this stage that developers make best of the knowledge about contexts and information requirements in the activity models and the reference models down to detailed levels, in order to construct a high-level architecture. The following figures (Figure 4.35 and Figure 4.36) are activity models and reference models related with machining operations.

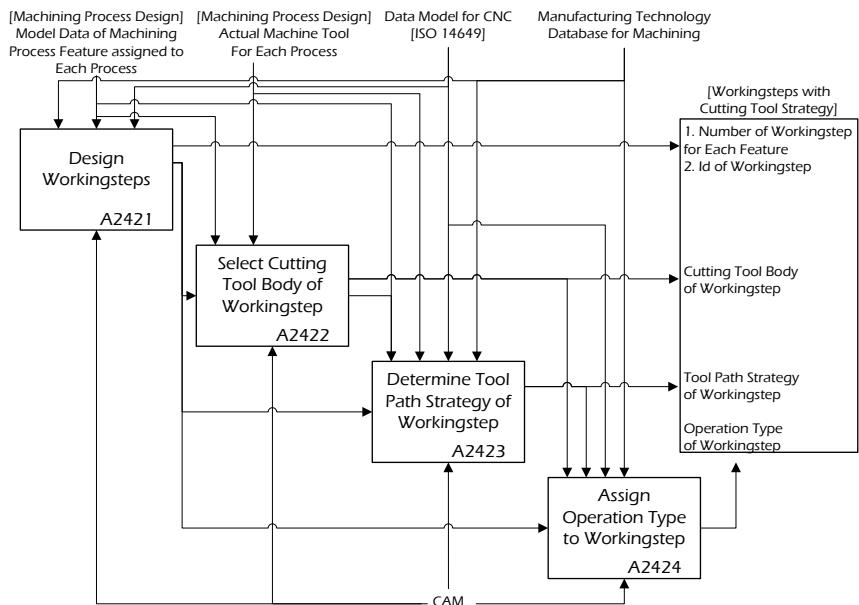


Figure 4.35 An activity model related with machining operations (author reproduction based on ISO 14649 STEP-NC).

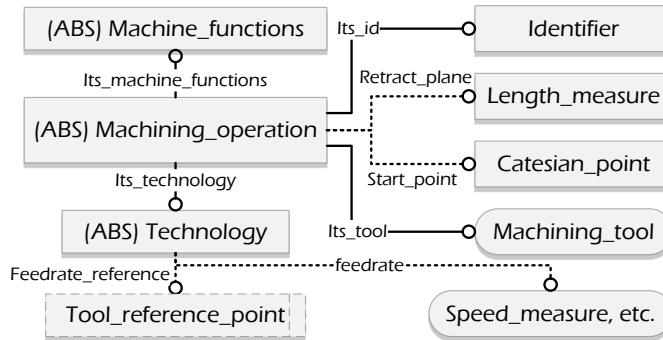


Figure 4.36 A reference model related with machining operations (author reproduction based on ISO 14649 STEP-NC).

This is also a stage where the implementers begin to invest effort on the model mapping process. This mapping process could be troublesome sometimes, because the relationship may not be so clear between data and functions when using information models with less consideration of implementation. Designers should adapt the models to the application scope and create the framework with suitable IA. Obviously, the activity models, the reference models and the use cases have different semantic granularity. Scenarios and requirements established in the previous stage can help to build semantic connection between the models. The provided activity model in the diagram (Figure 4.33) is an example of little explication on its relationship with reference models. Such cases require designers to devote more effort to establish the relationship according to the domain knowledge.

4.7.1 Use cases

Software architecture design requires a structured outline of forms, behaviors, styles and I/O methods, where all the views about functionalities, data structure, workflows of key paths and interface design can be connected. The use case is a methodology that provides such an outline for designers. The use case as a UML (Unified Modeling Language) modeling technique adopted in this study is text-based (mainly), structured and step-by-step descriptions about how a user interacts with a system to achieve a specific goal. Tasks are allocated to users or systems based on the scenarios and the requirements defined previously.

A key path workflow can be presented in a way of concrete use cases (Stone, et al., 2005) to expedite identification of important objects, attributes and actions. The specific goal of concrete use cases can be defined directly from the requirements elicited in the previous section. Work reengineering is performed based on the concrete use cases to clarify concepts in a subject domain. It can be delivered with any decoration method preferred by the designers to indicate relevant information. For instance, one important requirement is to specify a risk by inputting related assessment information (Table 4.1). Objects are highlighted by bold, attributes are highlighted by underlines and behaviors are ignored.

User action	System response
The user selects a failure mode	The system highlights presentation (e.g. the <u>name</u> and the <u>ID</u>) of the failure mode in a failure mode collection . The system displays the <u>related process</u> with <u>parameters</u> , <u>related features</u> in the geometry of the product model, <u>related resources</u> .
The user specifies one or more risk items with failure effects (for the <u>final product</u> or a certain <u>downstream process</u> , related to <u>severity</u>), failure causes (could be the failure mode of lower level steps, related to <u>occurrence</u>), detections (related to <u>undetectability</u>) and <u>descriptions</u>	The system displays real-time suggestions of <u>naming</u> and <u>values</u> of <u>similar failure effects</u> , failure causes and <u>detections</u> in previous PFMEA documents and <u>reliability records</u> of <u>similar process steps</u> .
The user finishes the specification	The system displays the <u>RPN</u> value and <u>diagrams</u> regarding the <u>input values</u> and RPN in a real-time report and puts highlight <u>colors</u> on <u>notable</u> RPN and SOD values according to default or customized criteria

Table 4.1 Working reengineering on a use case to specify risk assessment.

4.7.2 A design framework

The results of concrete use cases can be conveniently mapped to a design framework (Cooper, et al., 2014). Following the sequence specified in the context scenarios, an abstract presentation of how to organize the data elements is generated in a hierarchical manner. Key elements that may be reused by several scenarios should be highlighted. Generally categorizing the elements contributes to brief understanding of data structure for data modeling in the next stage, in terms of attributes, methods, inheritance and collections.

Two types of elements can be generated from the reengineered use cases: Functional elements and data elements (Cooper, et al., 2014). Each interaction (e.g. a row in Table 4.1) in the concrete use cases forms a function element. The decorated nouns are taken as data elements. The direct mapping result can be preliminary since there may be replications and unrevealed items in the use cases.

The design framework is a foundation to bridge design and construction and to create formal data models for construction. This thesis uses a tree table (can be easily converted to a mind map) to illustrate the design framework with established connections between data elements and function elements of the design framework (see an example Table 4.2). Knowledge gathered in the previous stages and in this stage is synthesized into a hierarchical pattern with at least four levels. Other types of forms are also feasible according to designers' preference such as indented lists or block diagrams. The four levels are needed to represent the hierarchy of the design framework:

1. Main scenarios
2. Requirements
3. Functional elements
4. Data elements

Note that some elements are repeated in different branches in the table tree because they are being reused in different scenarios. The element called "feature" is an example for the case of risk assessment since it is a

significant part to link process planning and product design. These replications indicate the possibility of integration of functions or scenarios. Still, the objective of the design framework is to identify the elements, so that the items at the first two levels can be simplified and combined to avoid repetition. For instance, in the following Table 4.2, there is a node called risk identification and specification which is a combination of two requirements specified in the previous stage. Another example of the design framework is for kinematic data exchange (Table 4.3), which share the same setting of levels, where the requirements for imports and exports are combined to eliminate some repetitions.

Scenario	Requirement	Function	Data	Attribute
Initialization	Team gathering	Record team personnel	personnel	Responsibility
			role	
		moderator		
		Record time & place	time	
			place	
	Document preparation	Load core process data	Process models	features
				Process hierarchy
		Gather and integrate models	Related product models	features
				geometry
				General property
				classification
		Related product line models		
	Criteria setting	Specify criteria	Previous risk models	Name & values
				Criteria in previous risk models

		Specify criticality presentation methods		
Analysis	Process item view	View process data	operation	Parameters instantiation Type
			Process sequence	
			feature	GD&T Instantiation type
			rationale	
			View product data	geometry classification rationale
		View resource data	resource	Machine tools line workpiece fixture staff
			Integrate views	Feature view Process view Operation view Risk view
			Display a selected process or a selected operation	
			Display editor of a risk item	Risk item Failure mode Failure effect Failure cause detection RPN value
			Suggest possible values	suggestions Naming & values reasoning
		Display real-time criticality	Reuse data in synthesis	
synthesis			diagram	

	Criticality synthesis	Display graphical criticality report	Significant values		
	counter-measure	Display a selected risk			
		Specify countermeasure	countermeasure	Action description	
				personnel	
				deadline	
	export	Export report	Standardized model		
			Information for control plans and work instructions		
		Schedule for evaluation			
	evaluation	trigger	Reuse initialization setting		
			Visualize triggers	Changes in documents	Product models
					Resource models
					Process models
			Countermeasure related		
	Update exported	Export updated report	risk		
			countermeasures		

Table 4.2 Design framework of risk assessment.

Scenario	Requirement	Function	Data	Attribute
Initialization	Model verification	Identify geometric models	component	Id
				placement
				parent
		Identify kinematic models	link	component
				name
			joint	name
				type
				Link [2]
				Placement [2]
				Limit [2]

			Base	Base link
	Establish link-age	link		
Kinematic chain	Select links	component		
		link		
	Specify links	link		
Dataset I/O	Function selection configuration	General feature		
		Holistically display the function		
		Specify name and location	File name	
			location	
		Specify type	Model type	
	execution	Extension spell		
		Specify whether to display chain info		
	Export /import as specification	Progress no-		
		tification		

Table 4.3 Design framework of kinematic data exchange.

A fifth level may be involved to provide detailed description about attributes or properties of the data elements. For information architects, the data elements and the attributes are important resources. In IA, metadata, i.e. “information about information”, was used to define data elements as the basis of all organizational systems (Wodtke and Govella, 2009). Findability, as a system capacity requirement, is the focus of defining metadata in the web world, but metadata is also relevant for functional requirements. The way to describe and organize metadata is also useful for data elements. Wodtke and Govella (2009) defined three types of metadata: Intrinsic (what the thing is), administrative (the way to handle it) and descriptive (how a user describes it). For findability, descriptive information is most important, but it is not the case for different specific functional elements. For instance, descriptive information is of no use for an element that will not be directly interacted by users. Designers should carefully define data elements and attributes based on the functional needs.

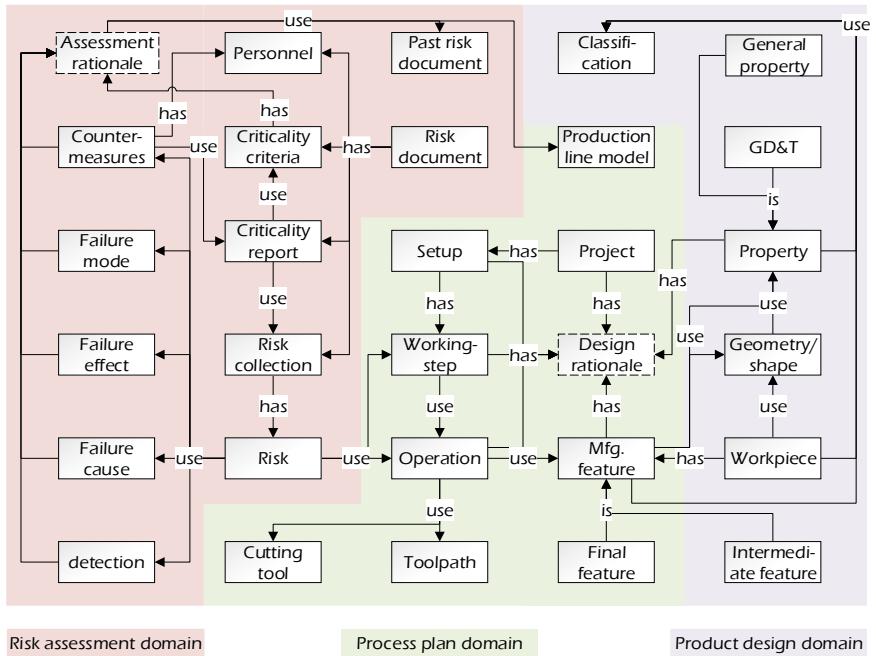


Figure 4.37 a high-level conceptual model for data elements.

The data elements for some functional elements (the third column of Table 4.2) are omitted since they are already referred to for the same requirements (the second column of Table 4.2). Note that there are still some data elements repeated from time to time for different types of scenarios or requirements and that it looks chaotic for this categorization manner of the data elements according to the functional elements. This thesis recommends a high-level concept model to diagram the data elements and relationships between one another (see an example in Figure 4.37). This example divides the data elements into three engineering domains: Risk assessment, process planning and product design. The division roughly gives holistic views in each engineering domain and it still shows the importance in practice of semantics of information models to support architecting system integration.

4.8 The abstraction stage

The design is detailed in the stage of abstraction, where software architecture for processes, objects and UIs (User Interfaces) are proposed for the implementers. Based on the previous stages (the general scope, the narrative conception and the formalized framework), this stage has been prepared for comprehensive implementation-oriented support. This thesis adopts UML class diagrams to illustrate process models and data models. UI design, often neglected by software engineers, is addressed in this stage as well.

In this stage, information model mapping is also performed between reference models and data schemas, which is the most difficult activity in the mapping. The below Figure 4.38 is a data schema excerpted from the ISO standard for the kinematic topology structure representation. This is a typical building block in the leaf level of a hierarchy to constitute an application protocol (e.g. STEP AP242) that is suitable for meaningful instantiation. It also reuses entities from other building blocks, e.g. a generic schema for representation structures (ISO, 2011) and another generic schema for topological representation (ISO, 2014c). As mentioned before (Section 2.2), this interpretation mechanism eases model development and improves extensibility of the information models. However, to a certain extent, it increases implementation complexity, because understanding this composable structure and performing information model mapping are not software practitioners' strength, compared with reading manuals, looking up API references and reading sample codes.

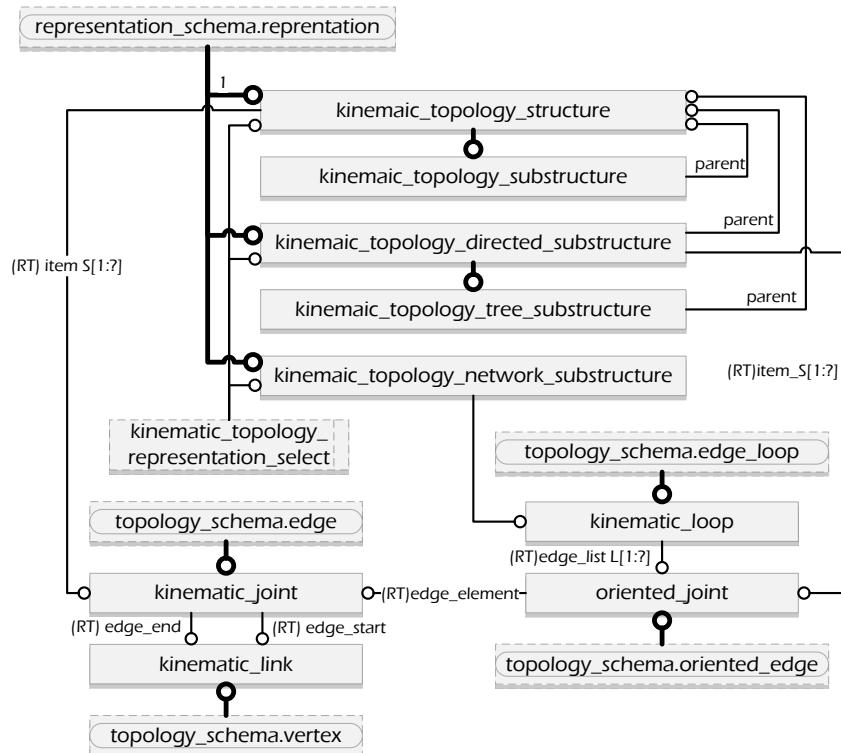


Figure 4.38 An excerpt of a data schema for STEP p105 ed2 (author reproduction based on ISO, 2014a).

This is also a stage to start to prepare a project-specific implementation model (Section 3.4), because relevant interrelated data elements with attributes have been stated in the design framework (Section 4.7.2). The data elements in the design framework suggest what should be encapsulated in an implementation model. The creation of an implementation model should be assisted by corresponding concepts in the data schema, e.g. ISO 10303 STEP p105 ed2 (ISO, 2014a). It also leads to a main part of a data model (Section 4.8.2) in the MVC (Model-View-Controller) pattern.

4.8.1 Process models

It is critical in this stage to keep the software architecture to follow principles of software engineering, IxD and designated implementation platforms. The basic aim to design software architecture is to reduce system complexity with widely applied principles such as high reusability, high cohesion of components, low coupling between components and single points of references. These principles are usually achieved by some generally defined techniques, such as layers, abstraction, encapsulation and modularization. There are patterns of software architectures to guide engineers to solve contextual problems for design systems, such as blackboards, pipes & filters, brokers, reflection and MVC.

As a blueprint for development of interactive systems with heavy content, the passive MVP as a variant of the MVC pattern is recommended for implementation of the information models. This selection is not a mandatory choice, especially when there is no HCI (Human-Computer Interaction) at all for some application contexts. With the passive MVP pattern (Sharan, 2015), the view is absolutely independent of the model, the model controls data and data logic related to the subject domain and the presenter controls logic about interface related to the interaction domain. To follow the convention defined in the classic MVC pattern, the presenter will still be called “Controller” in the rest of this thesis (Figure 4.39).

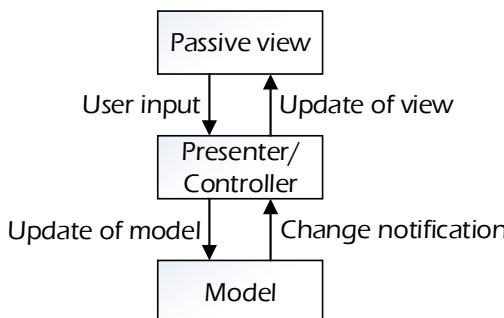


Figure 4.39 A passive MVP pattern.

MVC (Model-View-Controller, an architecture pattern)

MVC has been the most influential architecture pattern in actual use when UI is involved (Dix and Shabir, 2011). There are several understandings or revisions of the MVC pattern since the underlying concept (Figure 4.40) has been

addressed in the late 70s (Reenskaug, 1979). The basic principles used by all MVC variants are the same: 1) Above all, concerns of views (presentation) and of models (representation) are divided; 2) views are difficult to test, so no or fewer logics within view; 3) models have no idea about views, but controllers and views depend on models; 4) models are reusable for different views.

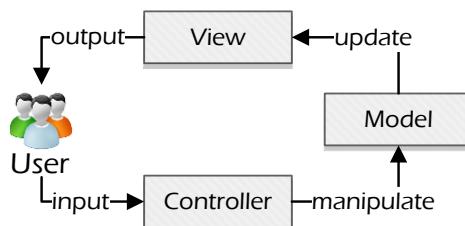
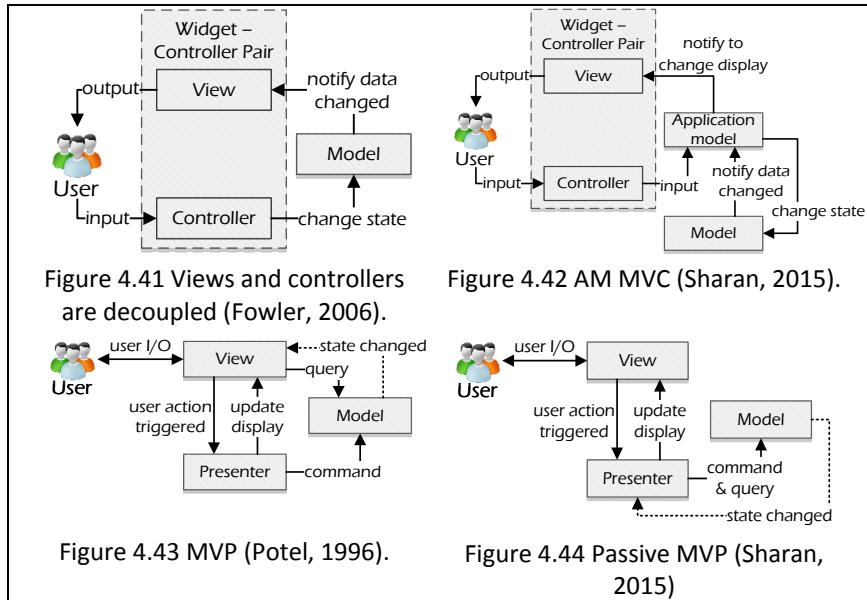


Figure 4.40 The classic concept of MVC.

Birth of GUI (Graphical User Interface) required change of this classic definition, because views and controllers were essentially interrelated in most implementations of GUI (Figure 4.41, Fowler, 2006). Decoupling them on purpose made nowhere to implement logics for the view. Hence, inventors of a revision of MVC inserted an application model between the model and the interface to handle these special situations, as Application Model MVC (AM MVC), in 1980s (Figure 4.42). However, in this revision, the controller is still redundant especially since the emerging of Microsoft Windows. MVP as a new version of the pattern redefined the relationship between the roles and became a widely-accepted solution (Figure 4.43, Potel, 1996), with its passive variant (Figure 4.44, Sharan, 2015). Nowadays, many famous patterns related with GUI, web and database adopt the idea of MVC, such as Web MVC (used by JSP Model 2, Struts and Ruby on Rails), MTV (used by Django), Web MVP (used by WebForms), MVVM (used by WPF and Silverlight).



The process design should make sure that all functional elements and data elements defined in the previous stage can be supported by a detailed software architecture. It is most important to pay attention to grouping and assignment of program logic in this step. There are many ways to document architectural design about processes. The UML interaction diagram is recommended by the 4+1 view model, but it can easily take too much space with an increasing number of components and interactions. Therefore, this thesis uses the normal flow chart with swim lanes, integrating the MVC pattern, to describe the processes (Figure 4.45). Although the interaction timing cannot be expressed explicitly, the diagram can still be understandable with clear definitions of components and data flows.

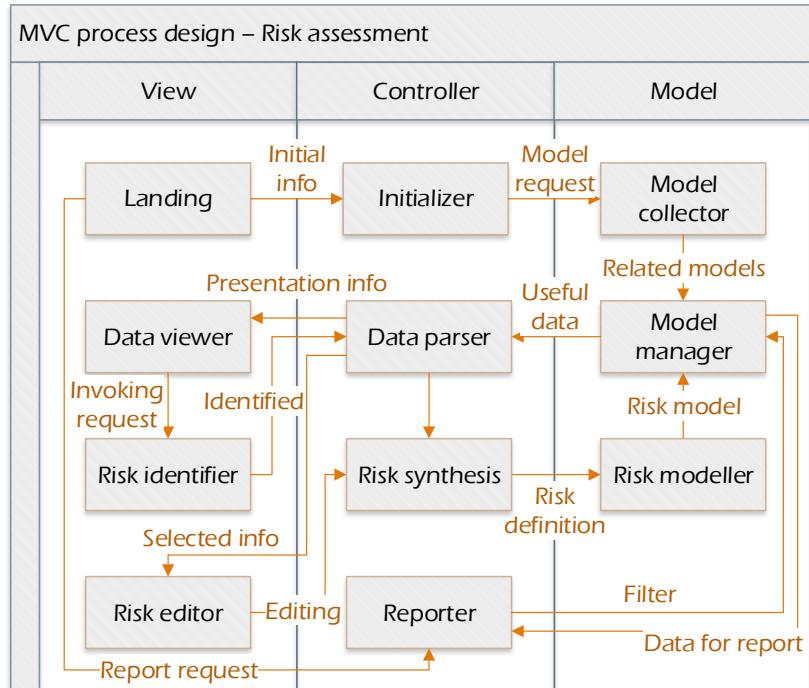


Figure 4.45 A flow chart to describe the process model.

4.8.2 Data models

To formally apply the data elements defined in the design framework to implementation activities, this stage uses UML class diagrams to convey a programming-oriented description of the data elements. The class diagrams for the data model represent concepts that are a part of the model module in the MVC pattern. In the below Figure 4.46, the designated data elements for the risk assessment example are mapped to a class diagram that will be a part of the model module.

In this stage, most decisions made by designers are related to attributes, data types, naming and relationships. It is possible that some data elements should be neglected temporarily. For instance, the criticality report is considered less related to the model but more related to the view, so it is better to place representations of concepts and logic of the criticality report in the controller module or the view module, rather than the model module.

Obviously, a full view of the software architecture design should not only include the classes for information, but also include those for functionalities that at least in both the controller module and the model module. Getters and setters are most common methods that might be involved in this stage. For instance, in the core class “RiskManager” in the exemplified diagram Figure 4.46. The getters and setters for other classes are omitted in this example.

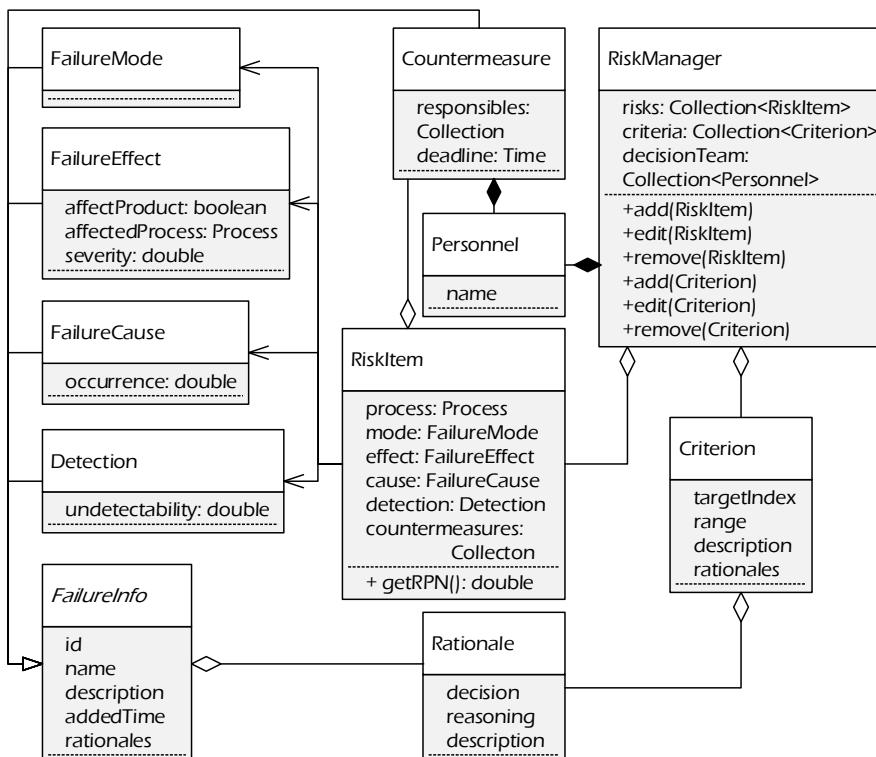


Figure 4.46 A class diagram to model the domain concepts.

4.8.3 Wireframe design

Today's IxD community has reached a common understanding that UI (User Interface) design is a systematic engineering project including the design activities previously defined in this chapter: User research, scenario

identification, requirement identification, IA, software architecture, etc. There will also be different levels of abstraction for UI design, e.g. sketch, wireframe, visual design and coding of interfaces (depending on platforms) which can be useful for different contexts. Ideally, most parts of the UI design should be performed by experts in different professions for different types of work products, e.g. interaction designers, software architects, visual designers and industrial designers. However, it is often that applications aiming at interoperability in industry are pressed for time so that such professional support cannot be found efficiently. Therefore, this thesis simplifies the UI design process by integrating the model driven approach, with the wireframe design as the last step of the abstraction stage. The wireframes are generated and indexed by the use cases from the previous stage.

Brown (2010) defined *wireframes* as “a simplified view of what content will appear on each screen of the final product, usually devoid of color, typographical styles, and images. Also known as schematics, blueprints”. Below are two examples of wireframes (Figure 4.47 and Figure 4.48) for two use cases: Identifying a specific risk item from an operation group and modifying risk items of a specific working step. Information regarding shapes, features and risks are displayed respectively and the relationship between risks, operations and processes are diagrammed as blocks to fit in users’ mental models. The wireframes highlight key functions and behaviors with cursors, which is not possible in reality with many cursors on the same screen (Figure 4.47). The wireframes avoid platform-specific details and aesthetic styles with a mostly black-and-white style which guarantees fast modification and iteration.

ARCHITECTING MODEL DRIVEN SYSTEM INTEGRATION IN PRODUCTION ENGINEERING

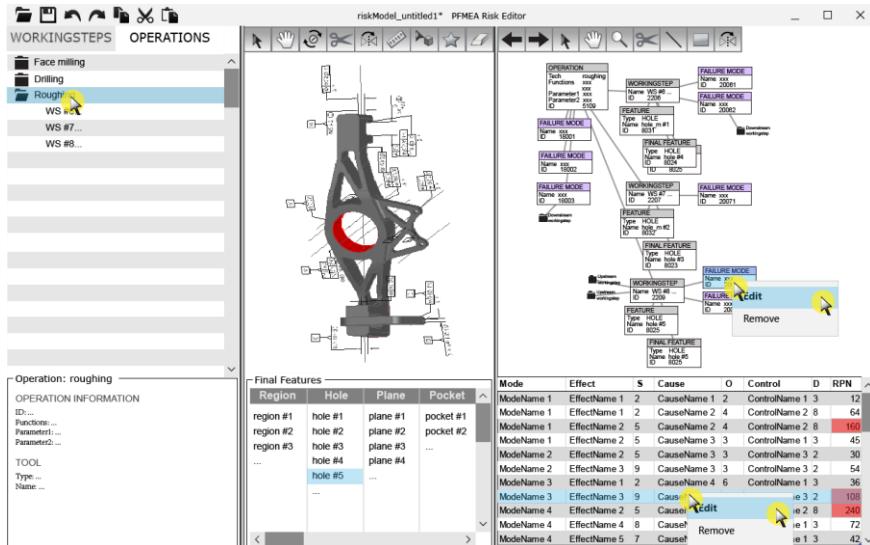


Figure 4.47 A wireframe design for identifying a specific risk item.

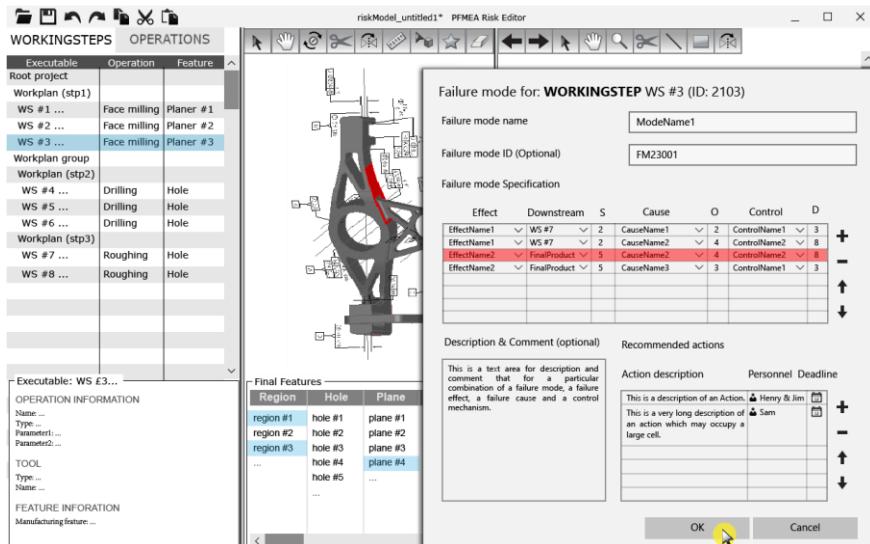


Figure 4.48 A wireframe design for specifying risk items.

4.9 The construction stage

Although a considerable amount of time has been spent in research, analysis and architecting, the application software cannot come into being without construction. Especially when it comes to system integration, developers should exploit potentials of existing applications and methodologies but bypass barriers caused by technical boundaries.

Based on a complex implementation context, the essential missions here in the construction stage are to minimize construction workloads caused by the interdisciplinary errands and to effectively adapt the developed application software in an existing context. The system integration is usually implemented in a practical engineering context. Most of deliveries are integrated applications rather than standalone ones, in order to reuse existing functionalities and data, or to be integrated into existing business processes. The involvement of multiple types of systems makes the implementation an interdisciplinary process performed by implementers with expertise in various domains.

In this stage, implementation details will be discussed in accordance with functional requirements, which may not be about system integration. Technically addition of functionalities is out of the scope of system integration, so these details are ignored in the previous stages. However, the possibility of demanding additional functionalities is inevitable in practice, due to the extensible modeling architecture. This fact significantly affects the landscape of the construction process for system integration. Therefore, the involvement of other functionalities will be investigated at this stage.

4.9.1 Program design

For this activity, two integration types in the course of model driven system integration should be generalized and discussed, according to implementation experiences of this research: 1) Integrate systems without necessity of implementing additional engineering; 2) the integration projects also demand additional functionalities due to missing links in the channel of interoperation.

The first integration type deals with a situation where existing CAx systems provide the required functionalities, as in the following interaction diagram in UML (Figure 4.49). With this design, users mainly interact with the existing CAx systems and the integrated application is developed only

to enhance the interoperability of the CAx. Typically, existing CAx applications provide existing interoperable solutions to some extent, e.g. all major CAD applications are capable to export geometric information in standardized interoperable formats such as STEP AP214, AP203, or AP242. The integration system is thus a supplement to the existing interoperable functionalities in such CAx applications, to avoid “reinventing wheels”. This integration design not only enable the developed applications fit into existing workflows, but also simplifies developers’ workloads by reusing existing functionalities.

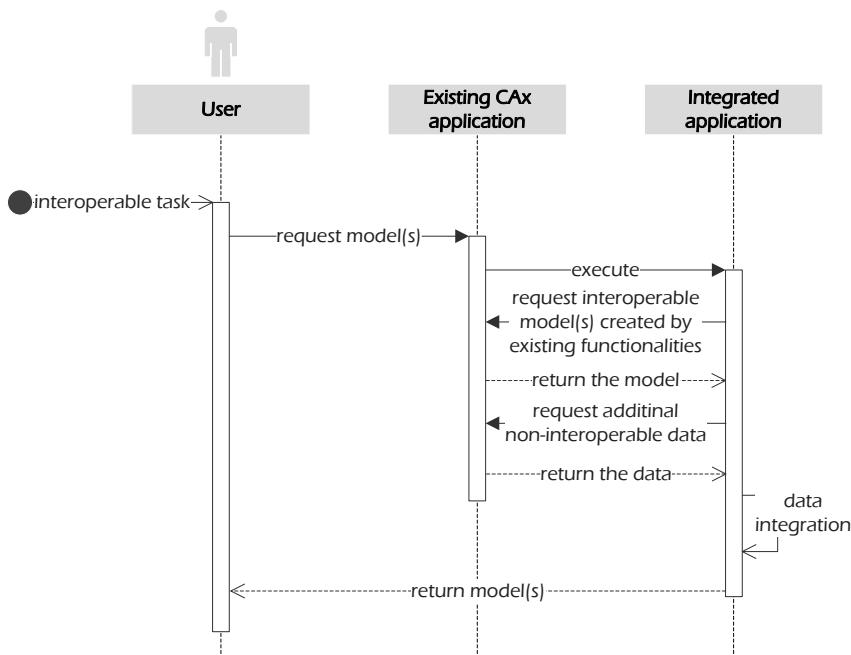


Figure 4.49 An interaction diagram for an implementation integrating with existing CAx applications.

The other integration type focuses on standalone implementation more, when there is a lack of both engineering functionalities and interoperability in existing systems. Interactive data editing contributes to a large

portion of the functional requirements on the implementation. The integration is essentially developed to add new functionalities, to enhance existing computerized information systems, or to automate a non-digitalized workflow. In such cases, a new workflow should be (and has been in the previous stages) invented and integrated with existing business processes. The information models are applied in the implementation to facilitate more than just interoperability, with broad application context and extensible modeling architecture. However, the system integration and the interoperability are still important because new functionalities need interpretable data input to avoid manual input. Accordingly, the implementers have more responsibility in developing new functionalities which result in obviously a lack of existing interoperable functionality for reusing. With this design, users mainly interact with the developed applications, with input of data sets exported from existing CAx applications (Figure 4.50).

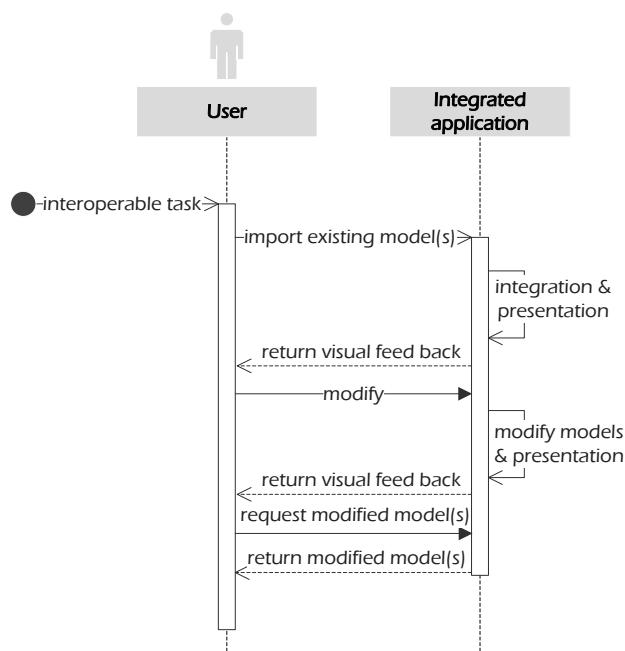


Figure 4.50 An interaction diagram for an implementation as standalone application integrated with users' existing workflow.

This system design enhances both the functionality and the interoperability and the information models are utilized to facilitate interpretation of the input data and the possibility for other systems to interpret the output data. Hence, the application is still an integrated system as illustrated, but integrated into a workflow rather than specific application software.

Although the manners to design the programs look different in both types, there are still some similarity. In both cases, the developed applications should extract relevant information, interpret models, integrate data and export interoperable data sets. Regarding manipulation of information models, the functionalities of both integration types are similar, apart from requests directly from users or not. Hence, reusing the MVC (Model-View-Controller) concepts adopted for the process modeling stage, behaviors of the implementation can be generalized as the illustrated program design in the following Figure 4.51.

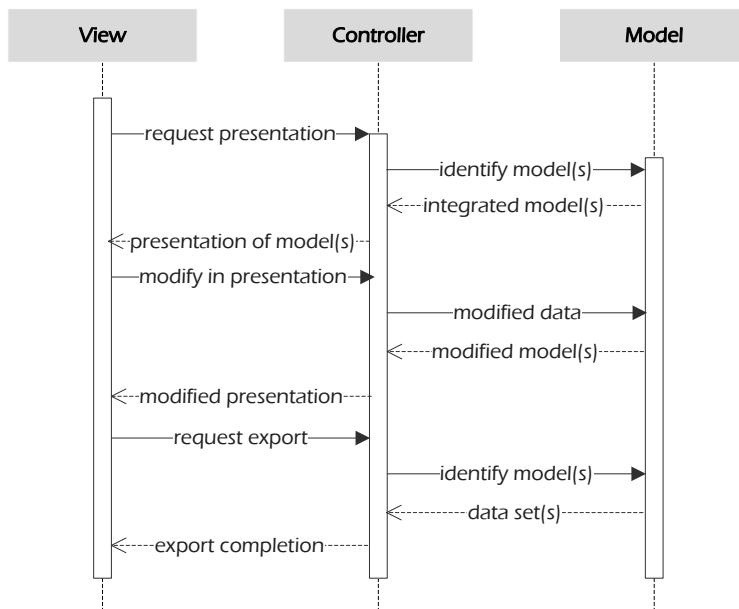


Figure 4.51 A general interaction diagram based on MVC for both types of implementations.

4.9.2 Programming and testing

As mentioned in Section 4.1, this chapter mainly focuses on delivering design decisions at an architectural level. As a result, the discussion about programming and testing, an activity close to implementation details, mainly focuses on synthesizing the deliveries from the previous chapter, i.e. the implementation models (Section 3.4) and the implementation architecture (Section 3.5), with the model component of the MVC pattern. The detailed construction of controllers and interfaces will be described to a limited extent.

The MVC pattern essentially divides focus points to cope with data, presentation and business logic. Different components have different responsibilities and delimited discoverable interfaces so that they are integrate-able. With the MVC pattern, the three subsystems with distinct functions and structure can be decoupled, implemented and maintained independently. This pattern should be fitted into a specific problem-solving context. The context for this scenario is the implementation context (Section 3.3), where the implementers implement applications according to user requirements in an application context.

It is only a well-designed API for data models that can satisfy the users' requirements. Integrated within the modeling architecture proposed in Section 3.5, the implementation models (Section 3.4) are ready for reuse in an MVC-based software architecture. The implementation model acts as a high-level API encapsulating the generic information models, revealing a core part of the information models and providing supportive functionalities to satisfy the needs of programmers. In the next chapter, an example of the implementation model and two cases using it will be introduced respectively.

Chapter 5

Case studies

While hackers can be very good at designing interfaces for other hackers, they tend to be poor at modeling the thought processes of the other 95% of the population.

- Eric S. Raymond

The case studies are mainly based on the appended papers, where projects were performed collaborating with industrial partners and act as major validations and demonstrations for this study. Moreover, many principles and experiences were inferred or generalized from the projects.

Section 5.1 is mostly based on the paper D for an implementation model. The delivery of the paper D was adopted in paper A, B, C and the licentiate dissertation (Li, 2013b). Section 5.2 describes demonstrations in paper A and E for kinematic data management. Section 5.3 describes demonstrations in paper B and C for cutting tool data management.

5.1 Realization of an implementation model: STEP Toolbox

The introduction of the STEP Toolbox was based on a need to reduce the workloads to implement the ISO 10303 STEP as a standard family for information models. An implementation model was developed to take into account interests of implementers in industry with limited knowledge on information modeling and other technical domains. Besides, there was a research need to rapidly implement new information standards and test the performance in system integration. Hence, simplification of implementation efforts to a level acceptable by practitioners in industry and academia was a major driver of the implementation models.

To develop the implementation model, investigation on the implementation context mattered. Traditionally, implementation of standards followed a unique procedure (Figure 5.1). At first a schema or an AP (Application Protocol) should be selected according to a match of application

contexts. Since an application context of a generic AP was often very broad, implementers always needed to select a relevant and semantically-closed part of the application context based on the description in the AAM (Application Activity Model) to implement, according to the context research of the intended implementation. Then, this selected context would be refined and concretized to information requirements by selecting a suitable part of ARM (Application Reference Model), according to application requirements. Alternatively, a recently-employed modularized modeling architecture required the selection of information requirements not performed on an AP ARM, but on the AMs (Application Modules) constituting the AP, e.g. STEP AP242 (ISO, 2014b). Based on either AP ARM or AM ARM, an AIM (Application Integrated Model) was hopefully generated as a valid data schema for instantiation. Then, implementers should choose a binding way to bind SDAI (Standard Data Access Interface) and AP schemas with a chosen programming language. Schema compilation may be triggered if early binding was chosen. Thus, an API, possibly integrating the data schema was ready for programming. However, a valid data set could not be guaranteed by either a compiled program or an exported data set. Validation was a must for each program to achieve interoperability.

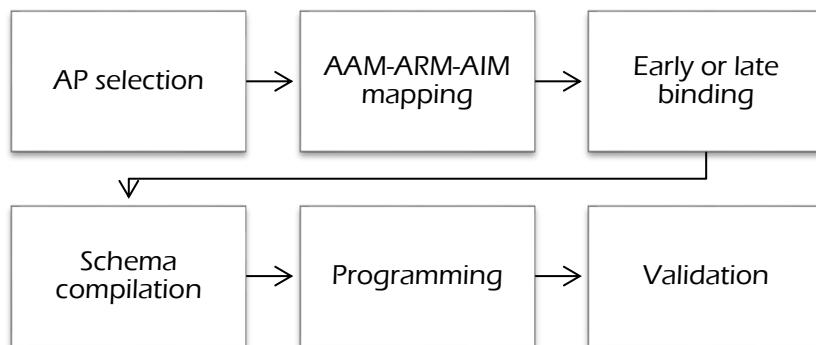


Figure 5.1 A unique process to implement STEP-based standards.

To sum up, this was not a convenient or intuitive process for either professional software engineers or end user programmers in manufactur-

ing industry. An API-based implementation process was often more preferred by practitioners. The API encapsulated the SDAI-based interface and the data schemas. Only concepts related with the subject domain were revealed for implementation, as well as domain oriented functionality support, e.g. coordinate translation. SDAI and model schemas would be hidden, but not be cancelled, because interoperability of the data sets was still a main concern. Besides, the original concepts would be revealed if necessary, for some functionalities beyond the scope of the API. With these design principles, the STEP Toolbox API (Figure 5.2) was proposed in the appended paper D.

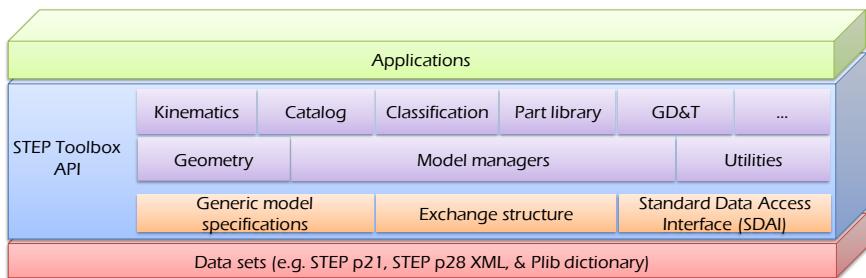


Figure 5.2 The modeling architecture using STEP Toolbox API.

As in the illustrated layered architecture, the STEP Toolbox API (the middle layer) acted as a connection between applications (the top layer) and data sets (the bottom layer). There were three sub-layers within API, the bottom sub-layer was the original modeling architecture that was useful for implementation, including the generic data schemas to specify semantics in an application context, the exchange structure to specify syntax and the SDAI to specify access interface. Based on the bottom sub-layer, a middle sub-layer was composed of three parts. The middle part, the model managers, was the central controller of all components in the middle sub-layer and the upper sub-layer and contained general operations for model manipulation sessions, such as initialization for different APs and finalization. The component named as geometry was to encapsulate geometric aspects of the data sets, since the geometric aspects were needed for almost all components in the upper sub-layer. The utilities contained functions to

manipulate information models that might not be provided by the standards, but generally necessary for implementation, e.g. coordinate translation. In the top sub-layer, components to represent different aspects of the information models were provided. These components were usually constituted by a manager and a set of classes directly representing relevant concepts, e.g. kinematic joints and kinematic links.

Class	Description
ClassificationManager	A manager to control PLib-based classification
ClassPLib	PLib-based external class for classification.
Component	A class to represent information of an occurrence
CSTransformer	A toolkit class to transform coordinate system
Dimension	Geometric dimension with tolerances
GeneralFeature	General design feature, can be classified
GeneralProperty	General property, usually not associated with specific geometric objects
Geometry	Super class for all geomtric elements, from components to points.
JarManager	Export the information about the entity types

Figure 5.3 Documented STEP Toolbox API by Javadoc.

With the layered architecture, an object-oriented API (see Figure 5.3 illustrating an exemplified documentation of the API) was provided as an intuitive way for the implementers. In the next two sections, utilization of this API in two application contexts will be introduced, integrated within the model driven implementation process (Chapter 4).

5.2 Kinematic data management

In this case, a project involving kinematic data management is studied with several demonstrations performed by different people with the STEP Toolbox API and the modeling architecture. Moreover, the implementation process introduced in the previous chapter is also integrated partly to enable efficient development.

The first demonstration was presented in the appended paper E. For all the demonstrations, the user study has been used as an example in Section 4.5 and been presented as a proto-persona illustrated in Figure 4.28. The problem could be stated as: For machine tools, there was a lack of a feasible solution to effectively automatically exchange kinematic data integrated with geometric models which was often designed in CAD systems but used in CAM systems; and the vision could be stated as: A system neutral solution was provided with enabling integrated applications for CAD-CAM systems and an exchange structure integrating kinematic data with geometric data. The solution was also detailed in Figure 4.31, where two implementations were introduced as the two enabling systems for interoperability. Then, a three-step context scenario has been provided in Section 4.6 to further elaborate the vision and the requirements for the kinematic data exchange was also presented in Figure 4.34. After the design framework presented in Table 4.3, a high-level process model before using the STEP Toolbox API could be illustrated as Figure 5.4 (see more description in section 5 of the appended paper E). It was definitely not an ideal solution since a lot of technical details about information modeling are involved. Hence, in Figure 5.5 the STEP Toolbox was recommended and included in the solution (see more description in section 6 of the appended paper D).

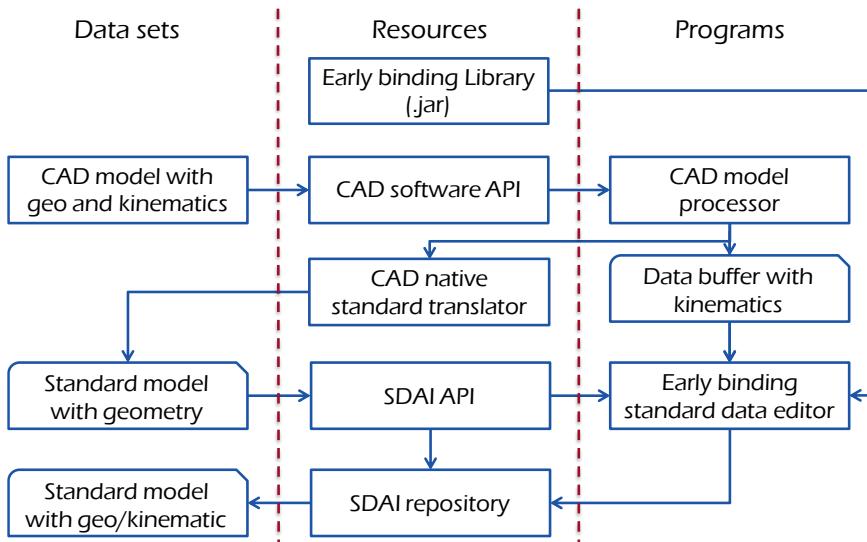


Figure 5.4 A flow chart for an integrated application to export a data set with full integration of kinematic data and geometric data.

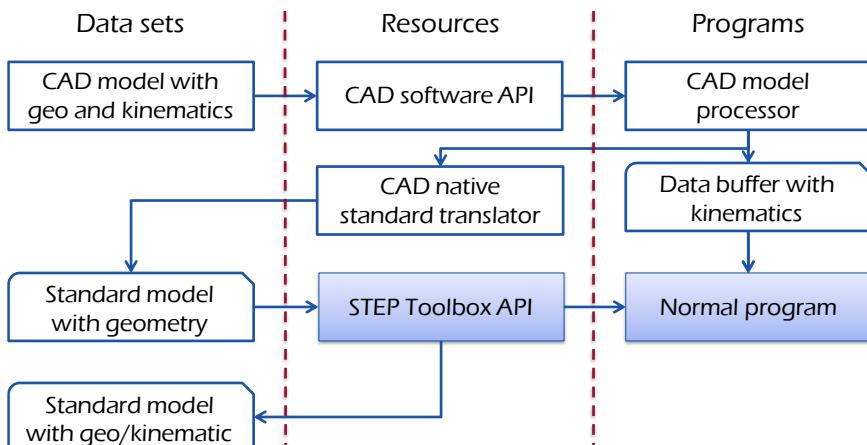


Figure 5.5 A process design enhanced with STEP Toolbox API.

Based on the solution illustrated in Figure 5.5, another demonstration was prepared in a collaboration project with industrial practitioners (section 6.2 of the appended paper D). In this project, the STEP Toolbox API was prepared as developed in Java. However, VBA (Visual Basic for Applications) was a preferred programming language to implement kinematic data I/O for a CATIA system in the partner organization (Figure 5.6). Hence, The STEP Toolbox API was compiled as an JAR (Java ARchive) file invoked by a VBA program and connecting with the VBA program with a temporary buffer file similar as the data buffer shown in Figure 5.5. Hence, the same process design was directly migrated to a new implementation context for a similar application context and the STEP Toolbox API as an implementation model was proven reusable in a different implementation context.

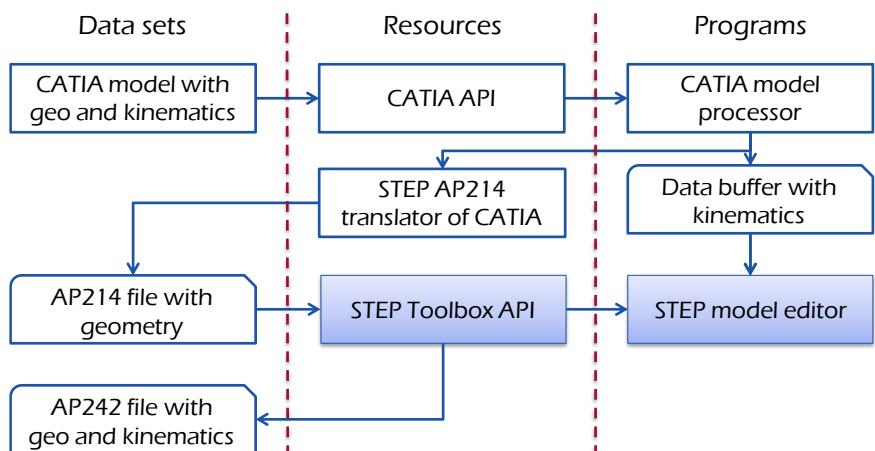


Figure 5.6 Process design of a kinematic data translator for CATIA.

Another project for kinematic error data management (paper A) was also prepared related with the kinematic data management. In this project, new types of representation for kinematic errors were involved, in terms of component errors and location errors. These new concepts demonstrated the high extensibility of the generic information model, i.e. STEP AP242 (ISO, 2014b) integrating the second edition of STEP p105 (ISO, 2014a). Based on this extension, an extended version (section 3 of paper A) was

also designed to revise the implementation model STEP Toolbox API (section 5.3 of paper D). This extended API made it fit into a new application context within a new implementation context which involved interaction with an ASME XML (ASME, 2008) information model.

5.3 Cutting tool data management

Besides kinematic data models mainly used for machine tools, classification mechanism (paper B) and categorization mechanism (paper C) of cutting tool product information were also major concerns in the projects related with this thesis. In particular, based on the classification mechanism, product catalogue exchange was enabled between cutting tool vendors with different catalogue structures and customers with different preferred visualization structure. This situation implied a level of interoperability higher than just semantic interoperability, but to a structural level in practical visualization. Integration with 3D geometric models was also a prioritized requirement which could be satisfied by the extensible modeling architecture of the ISO 10303 STEP standard family.

In the STEP standard family, STEP AP242 (ISO, 2014a), ISO 13584 PLib (Part Library, ISO, 2003a) and ISO 13399 (ISO, 2006a) for cutting tool definition were adopted as major facilitators for implementation. Specifically, the STEP AP242 was taken as an exclusive data schema for instantiation. The high extensibility of AP242 was demonstrated by extensions for cutting tool definition, classification and categorization.

For this demonstration, three integrated applications were proposed to enable an envisioned solution (Figure 4.30) and at least three types of data schemas needed to be prepared, i.e. representation definition of customer view hierarchy, representation definition of catalogue hierarchy and representation definition of catalogue classification. Note that the classification was applied on product classes in paper C rather than on specific products as in paper B. Then, mapping mechanisms were specified between the defined hierarchies, with the support of a standardized cutting tool dictionary defined in PLib (Parts Library, ISO 13584) by ISO 13399 to bridge vendor hierarchies and the customer view hierarchies (Figure 5.7). As a result, three integrated applications implemented the mapping mechanisms (see paper C for more elaborations of the mapping principles and the integration design).

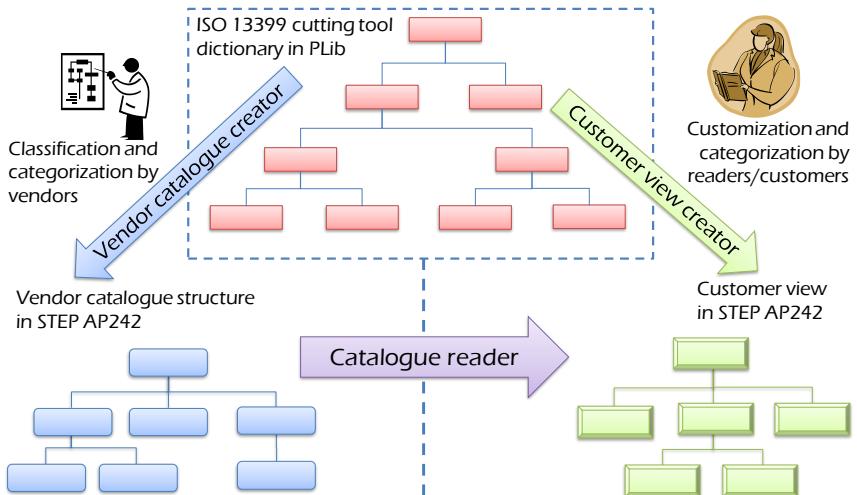


Figure 5.7 Mapping of hierarchy.

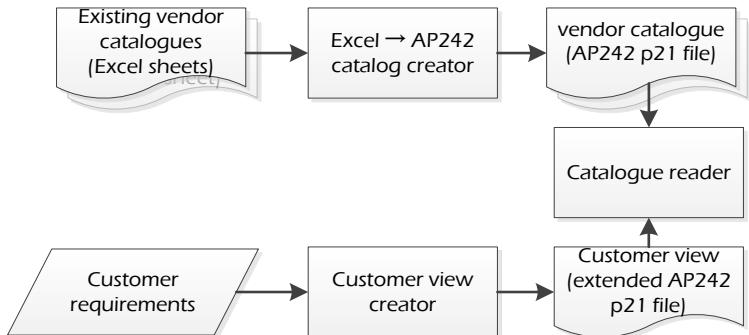


Figure 5.8 Prototypes for catalogue data exchange.

Three prototypes were developed to make use of the defined data schemas and the mapping mechanisms, i.e. an Excel-AP242 catalog creator, a catalogue reader and a customer view creator (Figure 5.8). Examples of vendor catalogues, provided by project partners, in MS Excel spreadsheets were taken directly as a data source. The customer view creator was used to specify preferred visualization structure to view cutting tool product information from multiple vendors. The catalogue reader processed outputs

from the previous two applications and created desired visualization of the catalogue data for the users.

Because of the design of multiple integrated applications, prototype development of this case was at a low detailing level. Most stages of the implementation process specified in Chapter 4 were not exactly followed. The deliveries were used to reveal principles to utilize the extended data schema based on STEP AP242 and other relevant schemas. Still, an implementation model was developed for cutting tool product information representation and classification representation, as depicted in section 3.1 of paper B for infrastructures of this case study. Besides, the implementation model was extended for catalogue data management to facilitate the prototypes in paper C.

Chapter 6

Conclusions

Essentially, all models are wrong, but some are useful.

- George E. P. Box

The model driven approach exploits potential of information models as a basis for system integration and as a key enabler for product realization in an engineering context. This approach needs to be implemented into application software with standardized generic information models in a designated context. Implementation is a bridge to connect information models and application contexts of using the information models.

However, implementation has always been an inherently difficult task (Brooks, 1987). In particular, interdisciplinary efforts are demanded for system integration for CAx systems in manufacturing industry. Implementers may experience inefficiency due to a lack of knowledge and skills. Accordingly, application users suffer from inadequate interoperability. This thesis tackles the problem with two strategies: Enhancing modeling architectures and streamlining implementation process. Both strategies are targeted at efficiency improvement, i.e. to reduce resources expanded to effectively implement the model driven system integration.

For the first strategy (Chapter 3), based on an implementation context, this thesis specifies an implementation model and a modeling architecture to integrate the implementation model. With this solution, the efficiency of using and reusing information models is significantly increased. Namely, implementers can skip time-consuming steps required by the traditional modeling architectures, e.g. model mapping, binding and validation. Meanwhile, the solution keeps the existing way to capture information in a complete, valid and extensible fashion. This solution has been validated by instantly increased implementation efficiency for industrial project partners and in-house research work, which also validates Hypothesis I:

HYPOTHESIS I

Introducing implementation contexts and implementation models in a modeling architecture improves efficiency of system integration.

The second strategy (Chapter 4) is elaborated as an integrated model driven implementation process for system integration. Potential of standardized generic information models is made use of, for highly qualified semantic interoperability and for the similarity in procedure. Stages and activities, focusing on agile architectural decision making, are defined in the integrated implementation process. The process is underpinned by information models as a knowledge base rather than technical constraints. The implementation process has been utilized as an underlying procedural principle for demonstration development in several projects (Chapter 5). Hence, the integrated model driven implementation process with the case studies validates Hypothesis II:

HYPOTHESIS II

Facilitated with existing modeling architectures and HCD (Human-Centered Design) principles, a software implementation process enables effective use of information models for system integration.

This thesis extends the model driven approach to a contextual level in the form of model driven system integration. In Hypothesis I, a contextual layer is added to the traditional modeling architectures, where the model driven approach can be efficiently realized. In Hypothesis II, a contextual process is introduced to fully reuse the traditional modeling architectures, where information models act as a knowledge base to drive the implementation practices.

6.1 Discussion

New understanding of architecting information models

The introduction of two new concepts, implementation contexts and implementation models, provides new understanding of how to design a modeling architecture. These concepts acknowledge fundamental pragmatic difference between data schemas and data sets: The primary users of

data schemas are implementers, and the primary users of data sets are application users. Based on data schemas, implementers develop applications which help application users to access data sets (Figure 6.1). Hence, pragmatically usable data schemas are prerequisite of pragmatically usable data sets. Qualified data sets do not suggest qualified information models. This understanding cannot be clear if model developers stick to application contexts as the basis of modeling architectures. In this regard, model developers should design data schemas that are more usable for implementers to make the data sets more accessible for application users.

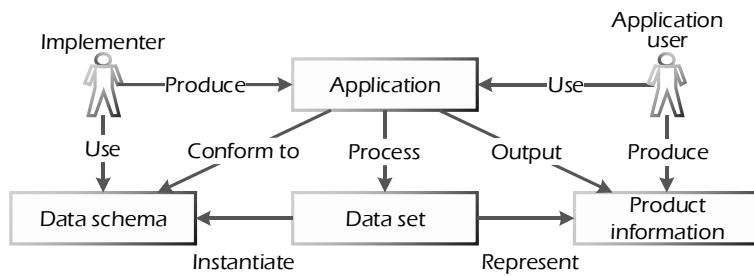


Figure 6.1 Implementers develop applications based on data schemas to make data sets accessible for application users.

Returning ownership of product data

A direct result of this study is returning ownership of product data to its true owners, the engineers. Engineering design activities could be performed with different systems by different people. Proprietary lock-in often costs huge frustration to reproduce data manually, or to spend substantial integration cost, where losses and errors are unpredictable. This thesis focuses on increasing performance of system integration for product realization based on standardized generic information models. The information models become a bridge between production engineering and software engineering, rather than a barrier. With efficient implementations, it is possible for industrial practitioners to be easily armed by integrated applications with holistic access to every corner of the data.

Avoiding the tool-driven manner

The model driven approach is a key enabler to help engineers focus on delivering the best design, without the delimitation by any software. For years, engineering applications have been so great tools for engineers that sometimes “it’s tempting for people to obsess over tools instead of what they’re going to do with those tools.” (Fried and Hansson, 2010) In most of current CAx software, there is an intrinsic template, representing a predefined application context implicitly or explicitly. The template not only draws a boundary for content, but also restricts engineering design procedures. Therefore, any application is likely to limit possibilities of engineering design solutions. In other words, the basic design principle “form follows function” (Sullivan, 1896) is violated. A good tool shall be invisible, that is, shall not limit the ways to accomplish user goals. The model driven approach is a novel solution which helps to integrate and collaborate the strengths of different systems. This study enhances the efficiency to apply the model driven approach, so that missing functionalities can be rapidly implemented and served. In this way, engineers will not be restricted by any individual system or any individual vendor.

Leading demonstrations

Using the results of this thesis, several leading demonstrations were accomplished. They contributed to the understanding of potential of information models and to the development of recently released new information standards, i.e. ISO 10303-242:2014 and ISO 10303-105:2014 ed2. The proposed implementation strategies also contributed to a few research projects, i.e. FBOP (Feature Based Operation Planning), DFBB (Digital factory building blocks) and MPQP (Model-driven Process and Quality Planning). In these projects, industrial partners were benefited of the work of this thesis for efficient implementation of system integration.

Being applicable besides system integration

The results of this study are also applicable for other scenarios that need information exchange, e.g. SOA (Service-Oriented Architecture) and cloud computing. A modeling architecture is for sure an infrastructure to support an SOA. Interoperability is the first quality attribute that should be taken

care of for SOA (O'Brien, et al., 2007) and it is the interchangeable information leading to interoperable services (Brownsword, et al., 2004). The information models, representing product information, have been proven, in this thesis, as an enabling leverage to interface general engineering applications and is capable to be extended for use in SOA. The system neutral information models guarantee a formalized contract that enables services discoverable, independent and reusable. Moreover, high agility of services can be assured by the implementation strategies proposed in this thesis, in that high portability of a formal contract is provided. Hence, the fundamental business goals of SOA can be assured, i.e. reduced TTM, integrated business processes and reusing legacy systems as services (O'Brien, et al., 2007). Cloud computing (Mell and Grance, 2011) can be seen as a specialized SOA with emphasis on networking and resource pooling and cloud manufacturing is a manufacturing version of cloud computing (Xu, 2012). Pallis (2010) identified three types XaaS (Everything as a Service) layers for different user groups. Although a top layer was put on a SaaS (Software as a Service) layer as "User front end", there are small but critical user groups relying on PaaS (Platform as a Service) and IaaS (Infrastructure as a Service). Compared with plain SOA, PaaS and IaaS are additional topics concerning development of cloud computing and cloud manufacturing. In particular, implementers are direct users of PaaS where effective system integration is required to implement SaaS. For information models as an infrastructure resource of PaaS, portability and extensibility are also crucial capabilities for efficient implementation.

6.2 Limitations

Necessarily high technological readiness

Development, deployment and utilization of computerized information systems are greatly affected by technological readiness of organizations and individuals. Although the model driven approach may reduce the workloads of immediate stakeholders to some extent, there is a progressive procedure to take effect. It is the job of engineering users and organizations to accept and embrace the technology. The content-rich systems (Section 2.5) require not only individual technological readiness (Parasuraman and Colby, 2015), but also an organizational level of technical infrastructures

in the supply chain. There should be adequate applications or services to provide functionalities to access useful digital information.

Object Oriented (OO) paradigm

In Chapter 4, this thesis adopts the Object Oriented (OO) paradigm to architect and construct the system. However, the presented methodology, which employs information models as a base for system integration, could be applied to any implementation paradigm. The two hypotheses of these thesis are targeted on practices in manufacturing industry and in IT industry respectively. The deliveries of engineering activity in both industries is shared: Documentation, after which the effort is handed to manufacturing team (Reeves, 2005). The information models play a unique role in both practices: As a base for innovation (Nielsen, 2003) to map from one reality (requirements) to another reality (designed artifacts). This role as in general engineering activities does not change whether the implementation paradigm is OO or not.

File-based integration scenarios

Architecting system integration is to explore an appropriate way to achieve interoperability between heterogeneous systems which can be technical units or business units. There are many styles to achieve system integration at an architectural level: File transferring, shared database, sockets, remote procedure call, messaging, etc. (Hohpe and Woolf, 2003) The style mostly discussed in this thesis is file transferring style which is the most common data sharing method (Roshen, 2009). However, the solution should not be limited to this one, because developers are faced with the same challenge no matter which style is selected. It is the challenge this thesis addresses: System integration in an unpredictably changeable environment. The issues are composed of 1) technical aspects: Unpredictable system performance, unpredictable system availability and evolving technologies and 2) business aspects: Changing business needs, changing regulations and changing intra- and inter-organizational structures. This thesis utilizes different techniques (e.g. standardization, information models, HCD, IA and MVC) based on the model driven approach to manage and minimize impacts from the constantly changing factors.

Limited context research

In-depth study of contexts, in terms of behavior patterns of users and systems for engineering design activities, is a missing area in both the domains of software engineering and HCD. This study identifies this issue and uses the implementation context to describe the problem space. A context is certainly useful, but it has an enormous scope. Technically a context should be a formal complete definition of a specific space-time where stakeholders are connected with a (probably envisioned) product. This definition is impossible to be derived completely and validly. On the other hand, there is a large amount of information in the definition that is irrelevant. Discerning what is relevant is the reason why designers have to research contexts, model users and analyze requirements. Although several industrial projects were involved in this study, there have been few opportunities to observe and learn from real application users and implementers.

6.3 Future work

The above limitations are certainly good starters for future work. However, there are potentially more important results that can be generated based on the following discussion. Especially, pragmatic concerns about high-level business requirements should be addressed from different perspectives. Knowledge from standards in other areas is useful for further research. Moreover, the standardized information models are widely used in building industry for interoperability as well. The fundamental idea is extremely similar to ISO 10303 STEP and the EXPRESS syntax is reused directly. However, remarkable work has been done for implementation-ready standardization and confusing terms identified in this thesis such as application contexts and information requirements are dropped. A cross-industry research could be beneficial for these two similar domains.

Integration aspects besides technology

The introduced solutions are potentially promising to overcome practical obstacles to adopt information standards from other perspectives more than technology. The direct driver of this thesis is a performance problem depicted technically, i.e. efficiency to implement system integration. However, “tools by themselves do not promote success; the proper use of the

tools does" (Putnik and Putnik, 2010). Business processes, culture and values are critical non-technological factors in relation with interoperability (Grilo and Jardim-Goncalves, 2010). Many researchers have investigated these factors, whose ideas could be borrowed here. For instance, Kiviniemi, et al. (2008) attributed delay of deploying BIM (Building Information Model) in AEC (Architecture, Engineering and Construction) industry to a triangle dilemma (Figure 6.2). Market demands, software supports and measured benefits together made a paradoxical loop that was hard to break. This is applicable for product realization in manufacturing industry as well. Tassey (1999) shared a similar viewpoint, based on general influences on business and concluded three causes of implementation failure: 1) Non-appropriability of benefits, 2) technical and market risks and 3) a lack of unbiased expertise. Gielingh (2008) also observed poor industrial uptake of STEP-related standards and imputed it to three causes: Low business motivation, low pragmatic readiness and legal aspects. It will be a new direction for developers and researchers on information standards to study how to take advantage of the result of this thesis for these high-level obstacles.

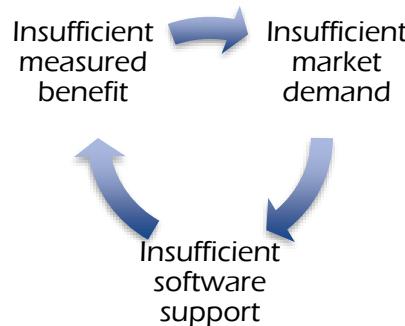


Figure 6.2 "Paradoxical loop" of implementing information models (author reproduction based on Kiviniemi, et al., 2008).

Future development of information standards

This study is also an incentive to raise concern of implementation readiness in future development of information standards. Usefulness of the

standards greatly depends on its usability in implementations and utilizations of applications and all applications start with implementations. There are several possibilities to increase implementation readiness formally:

- Develop standards with the focus on effective implementable solutions instead of effective standardization of concepts.
- Reuse and standardize the idea of conformance classes as functional subsets of schemas for specific implementation contexts.
- Keep a simplified implementer-oriented view in the modeling architectures, e.g. keep a small number of layers revealed; use the reference models as additional support rather than constraints.

Many standards have made efforts toward implementation readiness. In some APs (Application Protocols) of ISO 10303 STEP, conformance classes are prepared as subsets of standardized schemas suitable for implementation of specific scenarios. In particular, PLCSlib (OASIS, 2016) was specified to increase user experiences for implementers, based on a standard for Product Life Cycle Support (ISO 10303-239 PLCS, ISO, 2005). The IFC (Industry Foundation Classes, ISO, 2013) standard is also a decent comparable example of switched concern from concept standardization to a readily-implementable solution (Laakso and Kiviniemi, 2012). Moreover, an enormous amount of effort has been put into developing implementable IFC, e.g. an implementation guide directly for implementers (Liebich, 2009), IDM (Information Delivery Manual, ISO, 2016b) and MVD (Model View Definition) defined according to IDM. An MVD acts as additional (not mandatory) reference models but only specifies a useful subset of IFC for instantiation. This flat architecture facilitates comprehensible initial implementation with “useful minimum” scopes (Hietanen and Lehtinen, 2006). It does not require implementers translating business requirements from contexts to models via sophisticated mapping. The idea of MVD can be a helpful reference to implementation models or any information models that will be implementation oriented.

Implementation guidance

Domain-specific programmer-oriented implementation guidance is also needed for standardized information models. Formal data schemas are like codes in an API, for which the most important characteristic should be

“read like a book” (Gillis, 2016). Unfortunately, just like codes, data schemas are also written with unusual syntax and domain-specific semantics, but still should be readable by implementers. This fact demands model developers’ effort on proper implementation guidance:

- Fit with implementers’ existing workflows and mental models.
- Provide interpretation guidance on domain-specific semantics.
- Provide tailoring strategies based on implementation needs.

Implementers may be already familiar with implementing standards for system integration or interoperability, which implies the existing work flows and mental models that can guide implementation. Nonetheless, STEP and related standards are different from common data standards to achieve interoperability at a machine level or a syntactic level, e.g. HTML, SMTP, and TCP/IP. Standardized information models usually are targeted at a semantic level, which implies different implementation conventions. Semantic heterogeneity was a cause for the most difficult problems of data integration, which was considered unsolvable for decades (Ziegler and Ditttrich, 2004). However, there may be no generic solution, but domain-specific solutions have been devised in different standards, such as STEP and IFC. Implementation of the domain-specific solutions can be a strange experience for common implementers. For instance, partially implementing a standard, i.e. “embrace but omit”, to achieve syntactic interoperability may cause issues on standard integrity (Egyedi, 2007). On the other hand, information models standardized at a semantic level are meant to be tailored. An aggregate model for a holistic view and discipline models for specific-domain views are both useful for product data management (Figure 6.3, Van Berlo, et al., 2012). Hence, specialized guidance may be placed for implementers who are used to implement full web-based standards.

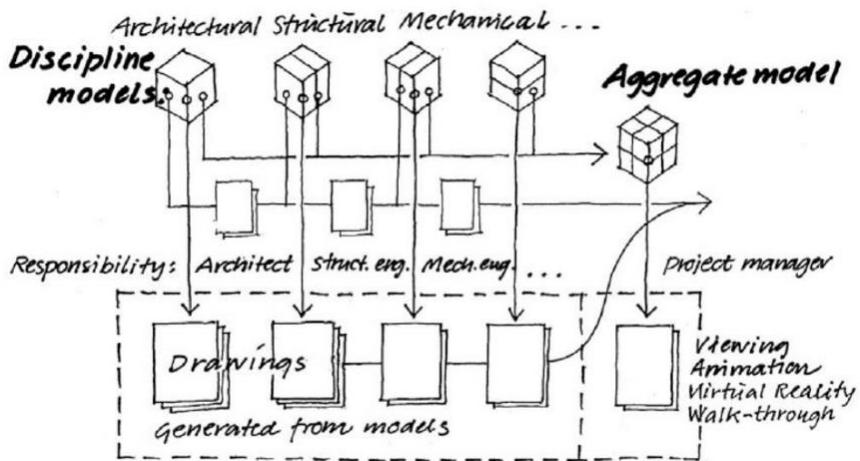


Figure 6.3 Discipline models and aggregate models in different stage of a product lifecycle (Van Berlo, et al., 2012).

HCD in information standards

This thesis might be an inspiration of human-centered information standards. “Technology is worthless - even dangerous - if we don't pay attention to the human aspects of both its use and its construction.” Gause and Weinberg (1990) pointed out a common pitfall faced by engineers in our times. BOTH use and construction of engineering applications are executed by human beings and the focused concern is always from human beings, implementers, users, designers, customers, etc. Standards that are used by implementers are also one type of these engineering applications which should be optimized with human centered design:

- Build decent information architectures to increase findability and ease of navigation.
- Design modeling architectures toward both end-user programmers and professional programmers.

Humankind and machinery excel in different aspects. The invention of machinery is driven by a desire to substitute intolerable operations of human intellect and to relieve “fatiguing monotony” (Babbage, 1822). When machinery is designed to interact with human beings, it should maximize performance of both parties, based on strengths and weaknesses of

the human information processing system (ISO, 1999). For most consumer products, traditional usability, in terms of effectiveness, efficiency and satisfaction, has become a “dissatisfier” (Jordan, 2000) that users take for granted. Nevertheless, the domain of information modeling has a different landscape: There is a huge gap regarding human-centered enhancement of information models. Standardized generic information models are not usually designed for usability, as an enabling system shall be intensively comprehended and used by implementers.

Definitions of relevant terms

Basic terms

Application software: "Software that is specific to the solution of an application problem" (ISO/IEC, 2015).

Architecture: Underlying structure of a system (Brand, S., 1995). "Fundamental concepts or properties of a system in its environment embodied in its elements, relationship, and in the principles of its design and evolution" (ISO/IEC/IEEE, 2011b).

Computerized information system: A "human-machine system that provides information to support the operational, managerial, analytic, and decision-making functions" based on the use of computers (Krogstie, 2012; Falkenberg, et al., 1998).

Context of use: "Users, tasks, equipment (hardware, software and materials) and the physical and social environments in which a product is used" (ISO, 2010b).

Data: A representation of information in a formal manner suitable for communication, interpretation, or processing by human beings or computers (ISO, 1994).

Information: "Facts, concepts, or instructions" (ISO, 1994). "Knowledge concerning objects, such as facts, events, things, processes, or ideas, including concepts, that within a certain context has a particular meaning" (ISO/IEC 2382:2015).

Information architecture: "<Human-centred> structure of an information space and the semantics for accessing required task objects, system objects and other information" (ISO/IEC, 2010, Section 2.5).

Integration architecture: Structure, relations and rationales of system components for system integration, decided at an architectural level.

Interactive system: "Combination of hardware, software and/or services that receives input from, and communicates output to, users" (ISO, 2010b).

Manufacturing: "The entirety of interrelated economic, technological and organizational measures directly connected with the processing/manufacturing of materials, i.e. all functions and activities directly contributing to the making of goods" (CIRP, 2004).

Model: A collection of representations to describe an existing or a future system (Blanchard and Fabrycky, 1990).

Requirement: "Statement which translates or expresses a need and its associated constraints and conditions" (ISO/IEC/IEEE, 2011a).

System integration: A process to progressively assemble "system components into the whole system" (ISO/IEC/IEEE, 2010).

Tool-driven manner: A problematic way to complete engineering goals greatly depending on tools in use with limited functions and content.

Terms about information models

Activity model: A model to represent an application context, in terms of processes, information flows, and functional requirements, which is usually to state scope of a data schema.

Application: "A group of one or more processes creating or using product data" (ISO, 1994).

Application context: An environment where product data is used in a specific application (a modified version based on ISO, 1994).

Application protocol (AP): A specification of a data schema satisfying an application context (a modified version based on ISO, 1994).

Data schema: An information model to specify representation of information in a particular context.

Data set: An information model instantiating a data schema or a conforming segment of a data schema.

Implementation context: An environment where implementers use an information model to implement an application.

Implementation model: An information model that encapsulates one or more data schemas, with additional supportive functions for data set manipulation, to satisfy information requirements in an implementation context.

Information model: "A formal model of a bounded set of facts, concepts or instructions to meet a specific requirement." (ISO, 1994)

Information requirement: Information that must be represented in a data schema.

Model driven: Keeping information in context, versionable, associative and retrievable in information models (Nyqvist, 2008); using information models to integrate product data from different sources, to create a productive environment and to enhance business competitiveness.

Modeling architecture: An architecture to specify development and use of an information model.

Model driven system integration: An integration practice to facilitate interoperability between disparate systems based on the model driven approach.

Product data: "A representation of information about a product in a formal manner suitable for communication, interpretation, or processing by human beings or by computers" (ISO, 1994), including data for lifecycle management, manufacturing, design, etc.

Reference model: An information model to represent information requirements of a specific application context (ISO, 1994).

Qualities pursued in this study

Effectiveness: "Accuracy and completeness with which users achieve specified goals" (ISO, 2010b).

Efficiency: "Resources expended in relation to the accuracy and completeness with which users achieve goals" (ISO, 2010b).

Extensibility: An ability to "continue to take advantage of new and innovative technique" (Al-Timimi and Mackrell, 1996). "The ease with which a system or component can be modified to increase its storage or functional capacity" (ISO/IEC/IEEE, 2010).

Interoperability: "The ability of two or more systems or components to exchange information and to use the information that has been exchanged" (ISO/IEC/IEEE, 2010).

Portability: An ability to "move data among applications" (Al-Timimi and Mackrell, 1996). "The ease with which a system or component can be transferred from one hardware or software environment to another" (ISO/IEC/IEEE, 2010).

Pragmatic interoperability: The ease to interpret meanings of exchanged messages by human audience.

Semantic interoperability: The ease to interpret meanings of exchanged messages by systems.

Syntactic interoperability: The ease to interpret formats of exchanged messages by systems.

Usability: "Extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" (ISO, 1998b).

Bibliography

- Abran, A., Moore, J. W., Bourque, P., Dupuis, R., & Tripp, L. (2004). Guide to the software engineering body of knowledge, 2004 version. IEEE Computer Society.
- Akin, Ö., & Lin, C. (1995). Design protocol data and novel design decisions. *Design Studies*, 16(2), 211-236.
- Alexander, C. (1979). The timeless way of building (Vol. 1). New York: Oxford University Press.
- Al-Timimi, K., & Mackrell, J. (1996). STEP: Towards open systems. Ann Arbor: CIMdata.
- Anderl, I. R., & Wasmer, D. I. A. (1997). Integration of product life cycle views on the basis of a shared conceptual information model. *Information Infrastructure Systems for Manufacturing*, 47-58. Boston: Springer.
- Annett, J. & Duncan, K. D. (1967). Task analysis and training design. *Occupational Psychology*, 41, 211-221.
- Arp, R., Smith, B., & Spear, A. D. (2015). Building ontologies with basic formal ontology. Cambridge: MIT Press.
- ASD (2009). Development of a convergent modular STEP application protocol based on AP 203 and AP214: STEP AP 242 – managed model based 3D engineering. ASD Strategic Standardization Group. [Online] <http://www.ap242.org/ap242-standard> [cited 2016 Oct.]
- ASME B5/TC56 (2008). ASME B5.59-2: Information technology for machine tools, part 2, data specification for properties of machine tools for milling and turning.
- Asuncion, C. H., & Van Sinderen, M. J. (2010). Pragmatic interoperability: A systematic review of published definitions. *Enterprise Architecture, Integration and Interoperability*, 164-175. Berlin Heidelberg: Springer.
- Babbage, C. (1822). A letter to Sir Humphry Davy, Bart on the application of machinery to the purpose of calculating and printing mathematical tables. London: J. Booth and Baldwin, Cradock and Joy.
- Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Merson, P., Nord, R., & Stafford, J. (2011). Documenting software architectures: Views and beyond, 2nd ed. Upper Saddle River: Addison-Wesley Professional.
- Bally, L., Brittan, J., & Wagner, K. H. (1977). A prototype approach to information system design and development. *Information & Management*, 1(1), 21-26.

- Bass, L., Clements, P., Kazman, R. (2012). Software architecture in practice, 3rd ed. Boston: Addison-Wesley.
- Bevan, N. (2013). Using the common industry format to document the context of use. International Conference on Human-Computer Interaction, 281-289. Berlin Heidelberg: Springer.
- Bevan, N., Carter, J., & Harker, S. (2015). ISO 9241-11 revised: what have we learnt about usability since 1998? International Conference on Human-Computer Interaction, 143-151. Springer International.
- Blanchard, B. S., & Fabrycky, W. J. (1990). Systems engineering and analysis (Vol. 4). New Jersey: Prentice Hall.
- Bloch, J. (2006). How to design a good API and why it matters. Companion to the 21st ACM SIGPLAN Symposium on Object-oriented Programming Systems, Languages, and Applications, 506-507. Chicago: ACM.
- Booch, G., Rumbaugh, J., & Jacobson, I., (2005). The unified modeling language user guide, 2nd ed. Boston: Addison-Wesley
- Box, G. E., & Draper, N. R. (1987). Empirical model-building and response surfaces, Vol. 424. New York: Wiley.
- Brand, S. (1995). How buildings learn: What happens after they're built. London: Penguin.
- Brhel, M., Meth, H., & Maedcher, A. (2015). Exploring principles of user-centered agile software development: A literature review. Information and Software Technology, 61, 163-181.
- Brooks, F. P. J. (1987). No silver bullet: Essence and accidents of software engineering. IEEE Computer, 20(4), 10-19.
- Brown, D. M. (2010). Communicating design: developing web site documentation for design and planning, 2nd ed. Berkeley: New Riders.
- Brownsword, L. L., Carney, D. J., Fisher, D., Lewis, G., & Meyers, C. (2004). Current perspectives on interoperability (No. CMU/SEI-2004-TR-009). Pittsburgh: Software Engineering Institute, Carnegie-Mellon University.
- Brunnermeier, S. B., & Martin, S. A. (2002). Interoperability costs in the US automotive supply chain. Supply Chain Management: An International Journal, 7(2), 71-82.
- Buley, L. (2013). The user experience team of one. New York: Rosenfeld Media.

- Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). A system of patterns: Pattern-oriented software architecture. New York: Wiley.
- Bush, V. (1945). As we may think. *The Atlantic Monthly*, 176(1), 101-108.
- Chen, D., Hedlind, M., von Euler-Chelpin, A., & Kjellberg, T. (2011). Using existing standards as a foundation for information related to factory layout Design. 21st CIRP Design Conference, 199-206.
- CIRP (2004). Dictionary of Production Engineering, Manufacturing Systems, Vol. III. Berlin and Heidelberg: Springer-Verlag.
- CMS (2008). Selecting a development approach. Office of Information Services, Centers for Medicare and Medicaid Services (CMS).
- CMU (2016). Community software architecture definitions. Software Engineering Institute, Carnegie Mellon University. [Online] <http://www.sei.cmu.edu/architecture/start/glossary/community.cfm>. [Cited 2016 Oct.]
- Constantine, L. L., & Lockwood, L. A. (1999). Software for use: A practical guide to the models and methods of usage-centered design. Pearson Education.
- Cooper, A., Reimann, R., Cronin, D., & Noessel, C. (2014). About face: The essentials of interaction design, 4th ed. Hoboken: John Wiley & Sons.
- Cusumano, M. A., & Selby, R. W. (1997). How Microsoft builds software. *Communications of the ACM*, 40(6), 53-61.
- David, P. A., & Greenstein, S. (1990). The economics of compatibility standards: An introduction to recent research. *Economics of Innovation and New Technology*, 1(1-2), 3-41.
- Dix, A. J., & Shabir, N. (2011). Human-computer interaction. Proctor, R. W., & Vu, K. L. (ed.), *Handbook of Human Factors in Web Design*, 2nd ed., 35-61. Boca Raton: CRC Press.
- DoD (2008). Systems engineering guide for systems of systems, Ver. 1.0. Office of the Deputy Under Secretary of Defense for Acquisition, Technology and Logistics, Washington, DC.
- Doran, G. T. (1981). There's a S.M.A.R.T. way to write management's goals and objectives. *Management Review*, 70 (11), 35-36.
- Egyedi, T. M. (2007). Experts on causes of incompatibility between standard-compliant products. *Enterprise interoperability*, 553-563. London: Springer.

- Falkenberg, E. D., Hesse, W., Lindgreen, P., Nilsson, B. E., Han Oei, J. L., Rolland, C., Stamper, R. K., Van Assche, F. J., Verrijn-Stuart, A. A. and Voss, K. (1998). FRISCO: A framework of information system concepts: The FRISCO report (WEB edition). International Federation for Information Processing (IFIP).
- Ford, N. (2008). The productive programmer. Sebastopol: O'Reilly Media, Inc.
- Ford, N. (2013). Why everyone (eventually) hates (or leaves) maven. [Online] http://nealford.com/memeagora/2013/01/22/why_everyone_eventually_hates_maven.html [cited 2016 Oct.]
- Fowler, M. (2006). GUI architectures. [Online] <http://martinfowler.com/eaaDev/uiArchs.html> [cited 2016 Oct.]
- Fricke, G. (1996). Successful individual approaches in engineering design. Research in Engineering Design, 8(3), 151-165.
- Fried, J., & Hansson, D. H. (2010). Rework. New York: Crown Business.
- Gallaher, M. P., O'Connor, A. C., Dettbarn Jr., J. L., & Gilday, L. T. (2004). Cost analysis of inadequate interoperability in the US capital facilities industry. National Institute of Standards and Technology (NIST).
- Garrett, J. J. (2010). Elements of user experience: User-centered design for the web and beyond, 2nd ed. Berkeley: New Riders.
- Gause, D. C., & Weinberg, G. M. (1990). Are your lights on? How to figure out what the problem really is. New York: Dorset House Pub.
- Gielingh, W. (2008). An assessment of the current state of product data technologies. Computer-Aided Design, 40(7), 750-759.
- Gillis, R. M. (2016). The New Frontier in Web API Programming. Lulu.com.
- Greene, R., Cushman, S., Cavanagh, C., Ramazani, J., Rouzer, P. F., Feinsod, H., Marno, D., & Slessarev, A. (2012). The Princeton encyclopedia of poetry and poetics. Princeton: Princeton University Press.
- Griffin, A., Lacetera, J., & Tolk, A. (2002). C4ISR/Sim technical reference model study group final report. Simulation Interoperability Workshop, Orlando, FL.
- Grilo, A., & Jardim-Goncalves, R. (2010). Value proposition on interoperability of BIM and collaborative working environments. Automation in Construction, 19(5), 522-530.
- Grudin, J. (1990). Interactive systems: Bridging the gaps between developers and users. IEEE Computer, 24(4), 59-69.

- Guarino N. Formal ontology and information systems. Proceedings of FOIS'98, 98(1998), 81-97.
- Hackos, J. T. (2002). What is an information model and why do you need one? The Gilbane Report, 10(1), 1-14.
- Hedlind, M. (2013). Model driven process planning for machining. Doctoral Thesis, School of Industrial Engineering and Management, KTH Royal Institute of Technology, Stockholm, Sweden.
- Hedlind, M., Lundgren, M., Archenti, A., Kjellberg, T., & Nicolescu, C. M. (2010). Manufacturing resource modelling for model driven operation planning. CIRP 2nd International Conference on Process Machine Interactions.
- Hedlind, M., & Kjellberg, T. (2014). Kinematical product specifications in engineering design. CIRP Annals-Manufacturing Technology, 63(1), 197-200.
- Hedlind, M., & Kjellberg, T. (2015). Design specifications with engineering terminology in a geometric context for CAD/CAM. CIRP Annals-Manufacturing Technology, 64(1), 169-172.
- Henning, M. (2009). API design matters. Commun. ACM, 52(5), 46-56.
- Hietanen, J., & Lehtinen, S. (2006). The useful minimum. Tampere University of Technology, Tampere.
- Hix, D., & Hartson, H. R. (1993). Developing user interfaces: Ensuring usability through product & process. New York: John Wiley & Sons, Inc.
- Hofmann, M. (2003). Essential preconditions for coupling model-based information systems. NATO Modeling and Simulation Group (NMSG) Conference on C3I and Modeling and Simulation Interoperability, Antalya, Turkey.
- Hohpe, G., & Woolf, B. (2003). Enterprise integration patterns: Designing, building, and deploying messaging solutions. Boston: Addison-Wesley.
- IEC (2006). IEC 60812:2006 Analysis techniques for system reliability - Procedure for failure mode and effects analysis (FMEA).
- IEEE (1998a). 1233-1998 — IEEE guide for developing system requirements specifications.
- IEEE (1998b). 1320.2-1998 — IEEE standard for conceptual modeling language — Syntax and semantics for IDEF1X97 (IDEFobject).

- INCOSE (2010). INCOSE-TP-2003-002-03.2 Systems engineering handbook, a guide for system life cycle processes and activities. SE Handbook Working Group, International Council on Systems Engineering (INCOSE).
- ISO (1994). ISO 10303-1:1994 Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles.
- ISO (1996). ISO 10303-105:1996 Industrial automation systems and integration — Product data representation and exchange — Part 105: Integrated application resource: Kinematics.
- ISO (1998a). ISO 10303-22:1998 Industrial automation systems and integration — Product data representation and exchange — Part 22: Implementation methods: Standard data access interface.
- ISO (1998b). ISO 9241-11:1998 Ergonomic requirements for office work with visual display terminals (VDTs) — Part 11: Guidance on usability.
- ISO (1999). Ergonomic requirements for the design of displays and control actuators -- Part 1: Human interactions with displays and control actuators.
- ISO (2003a). ISO 13584-24:2003 Industrial automation systems and integration — Parts library — Part 24: Logical resource: Logical model of supplier library.
- ISO (2003b). ISO 14649-1:2003 Industrial automation systems and integration — Physical device control — Data model for computerized numerical controllers — Part 1: Overview and fundamental principles.
- ISO (2004). ISO 10303-11:2004 Industrial automation systems and integration — Product data representation and exchange -- Part 11: Description methods: The EXPRESS language reference manual.
- ISO (2005). ISO 10303-239:2005 Industrial automation systems and integration -- Product data representation and exchange -- Part 239: Application protocol: Product life cycle support.
- ISO (2006a). ISO 13399-1:2006 Cutting tool data representation and exchange — Part 1: Overview, fundamental principles and general information model.
- ISO (2006b). ISO 13584-102:2006 Industrial automation systems and integration — Parts library — Part 102: View exchange protocol by ISO 10303 conforming specification.

- ISO (2007a). ISO 10303-238:2007 Industrial automation systems and integration — Product data representation and exchange — Part 238: Application protocol: Application interpreted model for computerized numerical controllers.
- ISO (2007b). ISO 10303-28:2007 Industrial automation systems and integration — Product data representation and exchange — Part 28: Implementation methods: XML representations of EXPRESS schemas and data, using XML schemas.
- ISO (2010a). ISO 10303-214:2010 Industrial automation systems and integration — Product data representation and exchange — Part 214: Application protocol: Core data for automotive mechanical design processes.
- ISO (2010b). ISO 9210-210:2010 Ergonomics of human — system interaction - Part 210: Human-centred design for interactive systems.
- ISO (2011). ISO 10303-43:2011 Industrial automation systems and integration — Product data representation and exchange Part 43: Integrated generic resource: Representation structures.
- ISO (2013). ISO 16739:2013 Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries.
- ISO (2014a). ISO 10303-105:2014 Industrial automation systems and integration — Product data representation and exchange — Part 105: Integrated application resource: Kinematics.
- ISO (2014b). 10303-242:2014 Industrial automation systems and integration — Product data representation and exchange — Part 242: Application protocol: Managed model-based 3D engineering.
- ISO (2014c). ISO 10303-42:2014 Industrial automation systems and integration — Product data representation and exchange Part 42: Integrated generic resource: Geometric and topological representation.
- ISO (2016a). ISO 10303-21:2016 Industrial automation systems and integration — Product data representation and exchange -- Part 21: Implementation methods: Clear text encoding of the exchange structure.
- ISO (2016b). ISO 29481-1:2016 Building information models — Information delivery manual — Part 1: Methodology and format.
- ISO (2016c). Standing document — New architecture for 10303 — Initial preliminary draft. ISO TC 184/SC 4 N8554. Date: 2016-07-26.
- ISO/IEC (2008). ISO/IEC 12207:2008 Systems and software engineering — Software life cycle processes.

- ISO/IEC (2010). ISO/IEC TR 25060:2010 Systems and software engineering — Systems and software product Quality Requirements and Evaluation (SQuaRE) — Common Industry Format (CIF) for usability: General framework for usability-related information.
- ISO/IEC (2014). ISO/IEC 25063:2014 Systems and software engineering — Systems and software product Quality Requirements and Evaluation (SQuaRE) — Common Industry Format (CIF) for usability: Context of use description.
- ISO/IEC (2015). ISO/IEC TR 13066-4:2015 Information technology — Interoperability with assistive technology (AT) — Part 4: Linux/UNIX graphical environments accessibility API.
- ISO/IEC/IEEE (2010). ISO/IEC/IEEE 24765:2010 Systems and software engineering — Vocabulary.
- ISO/IEC/IEEE (2011a). ISO/IEC/IEEE 29148:2011 Systems and software engineering — Life cycle processes — Requirements engineering.
- ISO/IEC/IEEE (2011b). ISO/IEC/IEEE 42010:2011 Systems and software engineering — Architecture description.
- ISO/IEC/IEEE (2015). ISO/IEC/IEEE 15288:2015 Systems and software engineering — System life cycle processes.
- Jacobson, D., Brail, G., & Woods, D. (2011). APIs: A strategy guide. Sebastopol: O'Reilly.
- Jacobson, I. (1992). Object oriented software engineering: a use case driven approach. New York: Addison-Wesley Professional.
- Jordan, P. W. (2000). Designing pleasurable products: An introduction to the new human factors. Philadelphia: Taylor & Francis Inc.
- Kazman, R. (2014). Software architecture in practice. New York: Addison-Wesley.
- Kiviniemi, A., Tarandi, V., Karlshøj, J., Bell, H., & Karud, O. J. (2008). Review of the development and implementation of IFC compatible BIM. Era-build Funding Organizations, Europe.
- Kjellberg, T., von Euler-Chelpin, A., Hedlind, M., Lundgren, M., Sivard, G., & Chen, D. (2009). The machine tool model—A core part of the digital factory. CIRP Annals-Manufacturing Technology, 58(1), 425-428.
- Knuuttila, T. (2011). Modelling and representing: An artefactual approach to model-based representation. Studies in History and Philosophy of Science Part A, 42(2), 262-271.

- Krogstie, J. (2012). Model-based development and evolution of information systems: A Quality Approach. Springer Science & Business Media.
- Kruchten, P. (1995). Architectural blueprints—The “4+ 1” view model of software architecture. *IEEE software*, 12(6), 42-50.
- Kruchten, P. (2003). The rational unified process: an introduction, 3rd ed. Boston: Addison-Wesley.
- Kuutti, K. (1995). Work processes: scenarios as a preliminary vocabulary. *Scenario-based design*, 19-36. New York: John Wiley & Sons.
- Laakso, M., & Kiviniemi, A. O. (2012). The IFC standard: A review of history, development, and standardization, information technology. *ITcon*, 17(9), 134-161.
- Lee, Y. T. (1999). Information modeling: From design to implementation. *Proceedings of the second world manufacturing congress*, 315-321. Canada/Switzerland: International Computer Science Conventions.
- Lewis, G. A., Morris, E., Simanta, S., & Wrage, L. (2008). Why standards are not enough to guarantee end-to-end interoperability. *The Seventh IEEE International Conference on Composition-Based Software Systems (ICCBSS 2008)*, 164-173.
- Li, Y., Hedlind, M., & Kjellberg, T. (2011). Implementation of kinematic mechanism data exchange based on STEP. *7th CIRP-Sponsored International Conference on Digital Enterprise Technology*, Athens, 152-159.
- Lieberman, H., Paternò, F., Klann, M., & Wulf, V. (2006). End-user development: An emerging paradigm. *End user development*, 1-8. Netherlands: Springer.
- Liebich, T. (2009). IFC 2x Edition 3 Model implementation guide version 2.0. BuildingSMART International Modeling Support Group.
- Lieb, H. H. (1971). On subdividing semiotic. *Pragmatics of Natural Languages*, 94-119. Netherlands: Springer.
- Lindland, O. I., Sindre, G., & Solvberg, A. (1994). Understanding quality in conceptual modeling. *IEEE software*, 11(2), 42-49.
- Liu, K. (2009). Pragmatic computing - semiotic perspective to web services. *Proc. 4th Int. Conf. on E-business and Telecommunications*, 23, 3-15. Berlin: Springer.
- Iivari, J., & Ervasti, I. (1994). User information satisfaction: IS implementability and effectiveness. *Information & Management*, 27(4), 205-220.

- Lundgren, M., Hedlind, M., & Kjellberg, T. (2016). Model driven manufacturing process design and managing quality. *Procedia CIRP*, 50, 299-304.
- Makris, S., Mourtzis, D., & Chryssolouris, G. (2014). Computer-aided manufacturing. Laperrire, L., & Reinhart, G., (ed.) *CIRP Encyclopedia of Production Engineering*, 254-266.
- Mannion, M., & Keepence, B. (1995). SMART requirements. *ACM SIGSOFT Software Engineering Notes*, 20(2), 42-47.
- Martin, R. C. (1996). The open-closed principle. More C++ gems, 97-112.
- Martin, R. A. (2005). A 'standards' foundation for interoperability. *Knowledge Sharing in the Integrated Enterprise*, 3-8. US: Springer.
- Matha, M. P. (2008). Object-oriented analysis and design using UML: An Introduction to unified process and design patterns. New Delhi: Prentice-Hall of India.
- Mayhew, D. (1999). The usability engineering lifecycle: A practitioner's handbook for user interface design. San Francisco: Morgan Kaufmann.
- McNeill, T., Gero, J. S., & Warren, J. (1998). Understanding conceptual electronic design using protocol analysis. *Research in Engineering Design*, 10(3), 129-140.
- Mell, P., & Grance, T. (2011). The NIST definition of cloud computing, Recommendations of the National Institute of Standards and Technology.
- MeLuhan, M. (1964). Understanding media: The extensions of man. New York: McGraw-Hill.
- Minsky, M. (1965). Matter, mind and models. Cambridge: Massachusetts Institute of Technology.
- Morris, C. W. (1938). Foundations of the Theory of Signs. Chicago: University of Chicago Press.
- Morris, C. W. (1939). Esthetics and the theory of signs. *The journal of Unified Science*, 8(1), 131-150.
- Mylopoulos, J., Chung, L., & Yu, E. (1999). From object-oriented to goal-oriented requirements analysis. *Communications of the ACM*, 42(1), 31-37.
- Nielsen, J. (2003). Information modeling of manufacturing processes: Information requirements for process planning in a concurrent engineering environment. Doctoral Thesis, School of Industrial Engineering and Management, KTH Royal Institute of Technology, Stockholm, Sweden.
- Norman, D. A. (2005). Emotional design: Why we love (or hate) everyday things. New York: Basic books.

- Nyqvist, O. (2008). Information management for cutting tools, Doctoral Thesis, School of Industrial Engineering and Management, KTH Royal Institute of Technology, Stockholm, Sweden.
- Nöth, W. (1990). *Handbook of semiotics*. Bloomington: Indiana University Press.
- O'Brien, L., Merson, P., & Bass, L. (2007). Quality attributes for service-oriented architectures. Proceedings of the international Workshop on Systems Development in SOA Environments, IEEE Computer Society.
- OASIS (2016). PLCSlip introduction. [Online] http://www.plcs.org/plcslip/plcslip_help/plcslip_content.html [cited 2016 Oct.]
- Ouksel, A. M., & Ahmed, I. (1999). Ontologies are not the panacea in data integration: A flexible coordinator to mediate context construction. *Ontologies and Databases*, 7-35. US: Springer.
- Ouksel, A. M., & Sheth, A. (1999). Semantic interoperability in global information systems. *ACM Sigmod Record*, 28(1), 5-12.
- Pallis, G. (2010). Cloud computing: The new frontier of internet computing. *IEEE Internet Computing*, 14(5), 70-73.
- Parasuraman, A., & Colby, C. L. (2015). An updated and streamlined technology readiness index TRI 2.0. *Journal of Service Research*, 18(1), 59-74.
- Parnas, D. L. (1994). Software aging. Proceedings of the 16th International Conference on Software Engineering, 279-287. IEEE Computer Society Press.
- Perry, D. E., & Wolf, A. L. (1992). Foundations for the study of software architecture. *ACM SIGSOFT Software Engineering Notes*, 17(4), 40-52.
- Peirce, C. S. (1907). Pragmatism. *The essential Peirce: selected philosophical writings* (1998), Vol. 2. Bloomington Indiana University Press.
- Pokraev, S., Quartel, D., Steen, M. W., & Reichert, M. (2007). Semantic service modeling: Enabling system interoperability. *Enterprise Interoperability*, 221-230. London: Springer.
- Posner, R. (1985). La syntactique, sa relation à la sémantique et la pragmatique, à la morphologie et la syntaxe, et à la syntagmatique et la paradigmatische. Parret, H., & Ruprecht, H. (ed.) *Exigences et Perspectives de la Sémiotique*, 1, 73-96. Amsterdam: Benjamins.
- Potel, M. (1996). MVP: Model-View-Presenter: The taligent programming model for C++ and Java. Taligent Inc.

- Putnik, G. D., & Putnik, Z. (2010). A semiotic framework for manufacturing systems integration -Part I: Generative integration model. International Journal of Computer Integrated Manufacturing - Semiotics-based Manufacturing System Integration, 23(8-9), 691-709.
- Redish, J. G. (2007). Expanding usability testing to evaluate complex systems. Journal of Usability Studies, 2(3), 102-111.
- Reenskaug, T. (1979). Thing-model-view-editor: An example from a planning system. [Online] <http://heim.ifi.uio.no/trygver/1979/mvc-1/1979-05-MVC.pdf> [cited 2016 Oct.]
- Reeves, J. (2005). Code as design: Three essays by J. Reeves. [Online] http://developerdotstar.com/mag/articles/reeves_design_main.html [cited 2016 Oct.]
- Rosén, J., (2010). Development of industrial information systems based on standards. Doctoral Thesis, School of Industrial Engineering and Management, KTH Royal Institute of Technology, Stockholm, Sweden.
- Roshen, W. (2009). SOA-Based Enterprise Integration: A Step-by-Step Guide to Services-based Application. New York: McGraw-Hill.
- Ross, D. T., Goodenough, J. B., & Irvine, C. A. (1975). Software engineering: process, principles, and goals. Computer, 5(8), 17-27.
- Rosson, M. B., & Carroll, J. M. (2009). Scenario based design. Sears, A., & Jacko, J. A. (ed.) Human-Computer Interaction: Development Process, 145-162. Boca Raton: CRC Press.
- Royce, W. W. (1970). Managing the development of large software systems. In proceedings of IEEE WESCON, 26(8), 1-9.
- Rozanski, N., & Woods, E. (2012). Software systems architecture: working with stakeholders using viewpoints and perspectives, 2nd, ed. Upper Saddle: Addison-Wesley.
- Rubin, K. S., & Goldberg, A. (1992). Object behavior analysis. Communications of the ACM, 35(9), 48-62.
- Rumbaugh, J., Jacobson, I., & Booch, G. (2004). Unified Modeling Language Reference Manual. Boston: Addison-Wesley.
- Schenck, D. A., & Wilson, P. R. (1994). Information modeling the EXPRESS way. New York: Oxford University Press.
- Shannon, C. E. (1948) A mathematical theory of communication. The Bell System Technical Journal, 27, 379–423 (Part I), 623–656 (Part II).

- Sharan, K. (2015). Learn JavaFX 8: Building user experience and interfaces with Java 8. Berkeley: Apress.
- Sharps, H., Rogers, Y., & Preece, J. (2007). Interaction design: Beyond human-computer interaction, 2nd ed. New York: John Wiley & Sons.
- Sherif, M. H. (2009). Handbook of enterprise integration. Boca Raton: CRC Press.
- Shneiderman, S. B., & Plaisant, C. (2004). Designing the user interface, 4th ed. Boston: Pearson/Addison Wesley.
- Sommerville, Ian (2011). Software engineering, 9th ed. Harlow: Pearson Education.
- Spencer, D. (2010). A practical guide to information architecture. Penarth: Five Simple Steps.
- Spinellis, D., & Gousios, G. (2009). Beautiful architecture: Leading thinkers reveal the hidden beauty in software design. Sebastopol: O'Reilly Media, Inc.
- Stephens, R. (2015). Beginning software engineering. Indianapolis: John Wiley & Sons.
- Stone, D., Jarrett, C., Woodroffe, M., & Minocha, S. (2005). User inter-face design and evaluation. Boston: Morgan Kaufmann.
- Sullivan, L. H. (1896). The tall office building artistically considered. Lippincott's Magazine, March 1896, 403–409.
- Syväjärvi, A., Stenvall, J., Harisalo, R., & Jurvansuu, H. (2005). The impact of information technology on human capacity, interprofessional practice and management. *Problems and Perspectives in Management*, 1(4), 82–95.
- Tassey, G. (1999). Interoperability cost analysis of the US automotive Supply Chain: Final Report. RTI Project, 7007-03.
- TED. (2014, June). Free software, free society: Richard Stallman at TEDxGeneva 2014 [Video file]. Retrieved from https://www.youtube.com/watch?v=Ag1AKl_2GM&t=1s.
- Tolk, A. (2003). Beyond technical interoperability-introducing a reference model for measures of merit for coalition interoperability. 8th CCRTS, National Defense University, Washington, D.C., June 2003.
- Tolk, A. (2004). Composable mission spaces and M&S repositories—applicability of open standards. Spring Simulation Interoperability Workshop, Washington, US.

- Tolk, A. (2006). What comes after the semantic web-pads implications for the dynamic web? Proceedings of the 20th Workshop on Principles of Advanced and Distributed Simulation. IEEE Computer Society.
- Tolk, A., & Muguiria, J. A. (2003). The levels of conceptual interoperability model. Proceedings of the 2003 Fall Simulation Interoperability Workshop, Vol. 7, 1-11.
- Tsichritzis, D., & Klug, A. (1978). The ANSI/X3/SPARC DBMS framework report of the study group on database management systems. *Information systems*, 3(3), 173-191.
- Van Berlo, L. A. H. M., Beetz, J., Bos, P., Hendriks, H., & Van Tongeren, R. C. J. (2012). Collaborative engineering with IFC: new insights and technology. 9th European Conference on Product and Process Modelling, Iceland.
- Van Lamsweerde, A. (2001). Goal-oriented requirements engineering: A guided tour. Proceedings. Fifth IEEE International Symposium on Requirements Engineering, 249-262. IEEE Computer Society.
- Vitruvius, P. (1914). The ten books on architecture. Cambridge: Harvard university press.
- Von Euler-Chelpin, A. (2008). Information modelling for the manufacturing system life cycle. Doctoral Thesis, School of Industrial Engineering and Management, KTH Royal Institute of Technology, Stockholm, Sweden.
- Wang, L., Shen, W., Xie, H., Neelamkavil, J., & Pardasani, A. (2002). Collaborative conceptual design—state of the art and future trends. *Computer-Aided Design*, 34(13), 981-996.
- Wodtke, & C., Govella, A. (2009). Information architecture: Blueprints for the Web, 2nd ed. Berkeley: New Riders.
- Xu, X. (2009). Integrating advanced computer-aided design, manufacturing, and numerical control: principles and implementations. Hershey: Information Science Reference.
- Xu, X. (2012). From cloud computing to cloud manufacturing. *Robotics and computer-integrated manufacturing*, 28(1), 75-86.
- Zeid, I. (1991). CAD/CAM theory and practice. New York: McGraw-Hill.
- Ziegler, P., & Dittrich, K. R. (2004). Three decades of data integration—all problems solved? Building the Information Society, 3-12. US: Springer.

Appended papers

Paper A

Li, Y., Hedlind, M. and Kjellberg, T. (2012a). Kinematic error modeling based on STEP AP242, *Proceedings of the 1st CIRP Sponsored Conference on Virtual Machining Process Technology*, Montreal.

Kinematic error modelling based on STEP AP242

Yujiang Li, Mikael Hedlind, Torsten Kjellberg

KTH Royal Institute of Technology, Production Engineering,

Brinellvägen 68, SE-10044 Stockholm, Sweden

yujiang.li@iip.kth.se

Abstract: Kinematic error is one of the major sources affecting geometric accuracy of machine tools motion. Component error and location error are common concepts to characterise kinematic error for both linear and rotation axes. With the second edition of STEP p105 currently under development, kinematic error can be associated directly to corresponding kinematic pairs. Based on the conceptual ontology model, a practical approach is introduced for standardized kinematic error modelling and implementation. Through integration with geometry, assembly, kinematics and classification, a developed application demonstrates and evaluates data representation and exchange for kinematic error. As STEP AP242 is a generic model for any product or manufacturing resource, additional modeling constraints are applied to achieve unambiguous data representation.

Keywords: kinematic error, modelling, data exchange, standardisation

1 INTRODUCTION

This paper presents a kinematic error modelling approach based on standards for data representation and exchange. Geometric accuracy of machine tool motion plays a major role in machining capability. The geometric error is significantly influenced by kinematic error [Schwenke et al., 2008], in terms of component error and location error. To achieve high accuracy in machining, a calculation methodology (multiplication of homogeneous transform matrices) for kinematic error compensation [Donmez et al., 1986] is commonly used. In application of virtual machine tool [Altintas et al., 2005] the representation of kinematic error is a key factor for realistic simulation. Historically, development of kinematic error measurement and analysis has been done with focus on machine tools, but based on general principles applicable for any product mechanism. This enables reuse of data schemas and computation methods for different applications. Although applications for kinematic error analysis are not as common as for nominal kinematic analysis at present, it can be expected that more and more commercial CAD/CAM/CNC systems will support it for more realistic virtual machining (see Figure 1). As what happened in the product design area, multiple systems with different functionalities may be used, often also in different organizations. Therefore, a standard for data representation and exchange of kinematic error will become as necessary as shape geometry to enable efficient information communication and sharing.

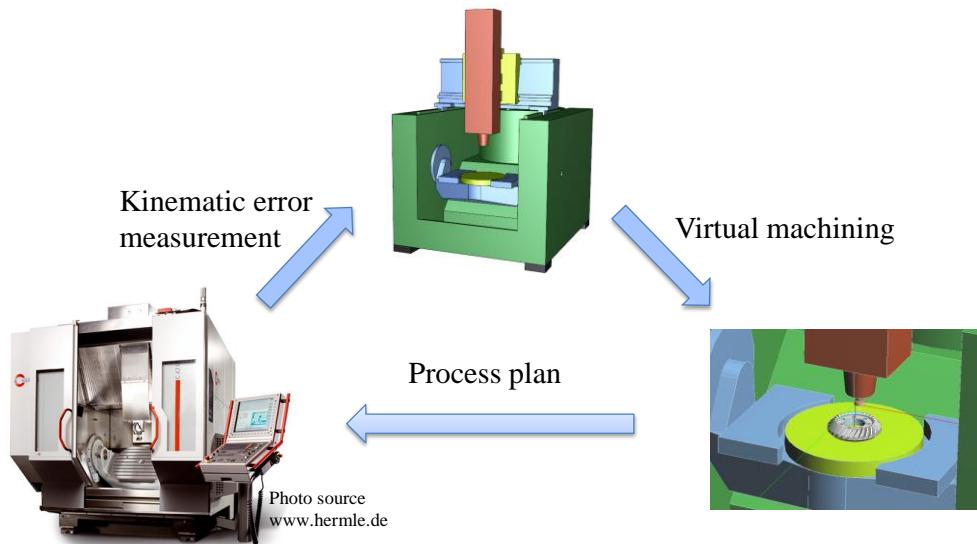


Figure 1; Machine tool modelling for realistic virtual machining

To support different utilisations in different domains, the standard data model for product and manufacturing resource should provide functionalities in a broad scope through the manufacturing industry [Euler-Chelpin, 2008], e.g. the kinematic error model should support measurement, analysis, prediction, and compensation. Therefore, the model should keep sharable concepts that can be interpretable in any domain, and it should be ready for fully data integration with other types of properties defined in different domains, e.g. geometry and kinematics. Therefore, a new standard, ISO/CD 10303-242 (STEP AP242, Managed model-based 3D engineering) [ISO, 2012], is chosen in this research for data modelling.

ISO 10303 STEP (STandard for the Exchange of Product data) is the international standard for industrial product data representation and exchange. Two application protocols of STEP, ISO 10303-203 (AP203, Configuration controlled 3D design of mechanical parts and assemblies) and ISO 10303-214 (AP214, Core data for automotive mechanical design processes), are implemented in all major CAD systems. ISO 10303-105:1996 (STEP p105 ed1, Kinematics) addresses kinematic mechanism representation. Currently AP214 is the only application protocol that uses p105 ed1. Till now the only valid application based on AP214 and p105 ed1 is implemented by Li et al. [2011]. However, due to the structure of p105 ed1, it is not possible to define associated properties, e.g. kinematic error, for existing kinematic elements.

At present, a project is ongoing to develop the new application protocol AP242 which is planned to unify and replace the AP203 and AP214. As an important part of AP242 development, standardised kinematic mechanism representation is being updated as the second edition of ISO 10303-105 (STEP p105 ed2, Kinematics). A major improvement of this edition is reusing the existing general representation construct defined in ISO 10303-43 (STEP p43, Representation structure), where the definitions and relationships of representation, items, and contexts are described [Hedlind et al.

2011]. Such structure provides possibility to define properties associated to different levels of kinematic elements: mechanism, kinematic pairs, and mechanism states.

Based on AP242, this paper presents a practical modelling approach to represent kinematic error as a property of kinematic elements. The general property representation structure defined in ISO 10303-41 (STEP p41, Fundamentals of product description and support) is used to represent the measured or calculated kinematic error information and the associations to corresponding instances defined in STEP p105 ed2. The data integration between geometry, kinematics, and kinematic error is evaluated and demonstrated by a prototype implementation, which indicates a comprehensive solution for data exchange between miscellaneous information systems.

2 KINEMATIC ERROR MODELLING

The kinematic error is divided into component error and location error, and the detailed definitions for both are different for rotational and linear axis [Schwenke et al., 2008]. Nominal kinematic pair frame placement as defined in STEP p105 (see Figure 2), is used to relate kinematic error concepts unambiguously in the context of STEP. Component error is the pair placement error from the respective nominal placement, and location error is defined as the average of contact frame placement error from its nominal placement, relative to the machine coordinate system.

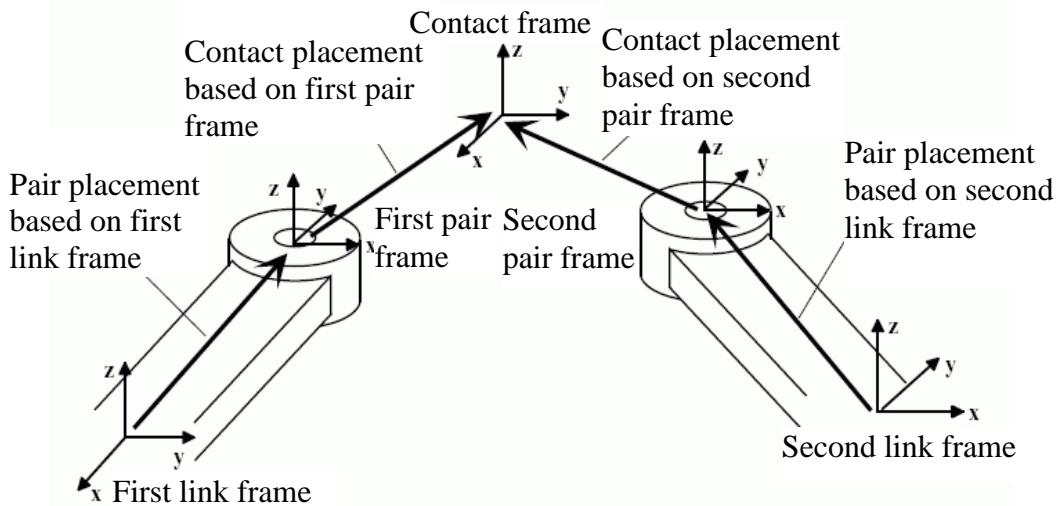


Figure 2; Placement relationship of a pair (ISO 10303-105)

Using the modelling principle defined by Kjellberg et al. [2009], an ontology model for the kinematic error concept is defined and applied on AP242. Figure 3 illustrates the kinematic error combined with geometry and kinematics ontology. Based on a unified modelling approach kinematic error can be modelled as kinematic pair property and represented using AP242 [Hedlind et al. 2010].

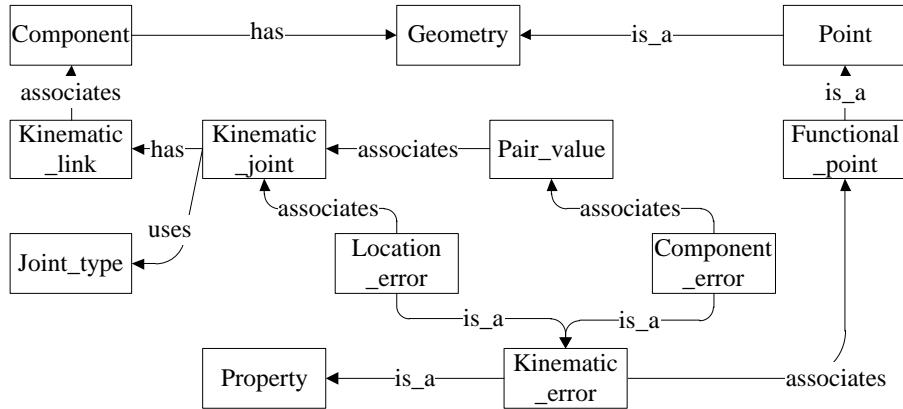


Figure 3; Kinematic error ontology

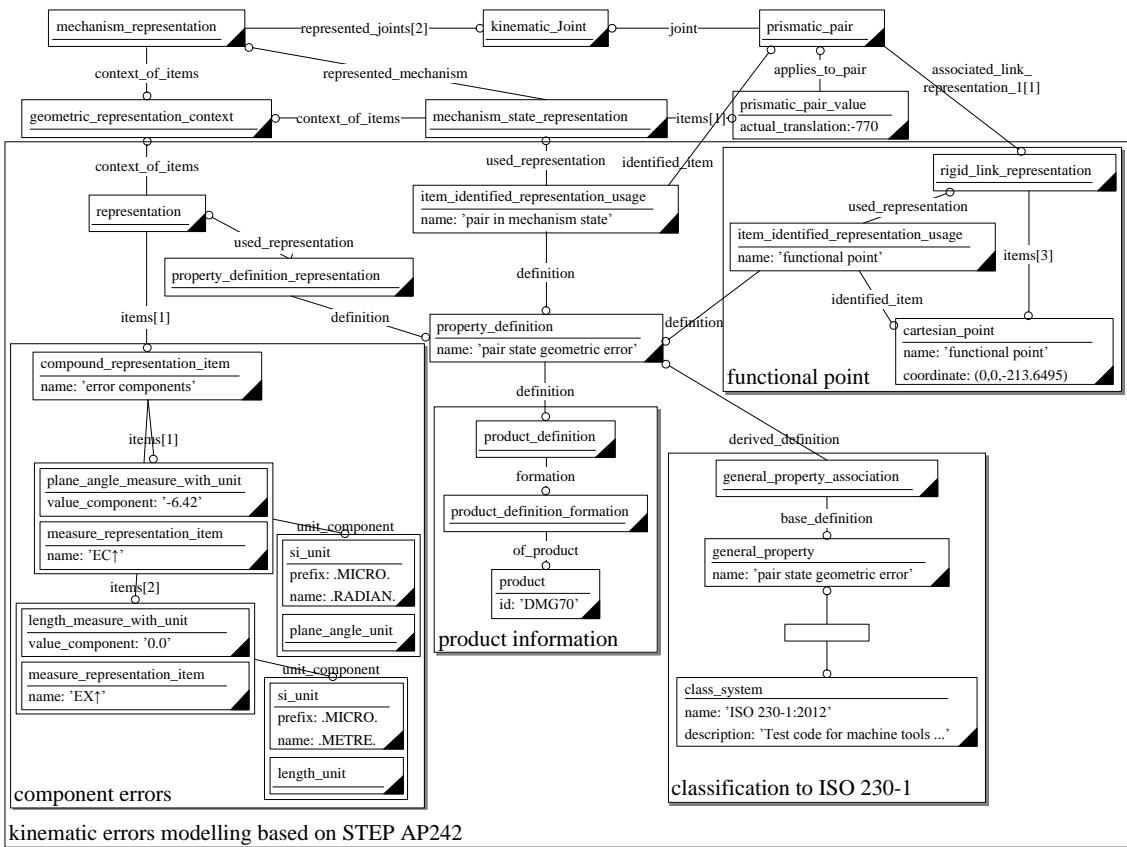


Figure 4; Component error modelling in AP242

With the presented ontology, the data model of kinematic error can be generated as classified properties in AP242. In Figure 4, a complete component error model is defined underlying the kinematic pair value model. A mechanism state is associated with a property named “pair state geometric error”. The representations of component error are collected and compounded under the context of one representation referred by

the property definition. The naming convention defined in ISO 230 is used as the name of each measurement value with unit, combined with the approach direction symbol \uparrow or \downarrow defined in the ISO 230-2. The product_definition of the assembly model is linked as the owner of the kinematic error property. A general property is assigned to the component error to declare the terminology and concepts defined in ISO 230-1:2012. The functional point is the location of the tool in the moving component when performing the straight line motion test as defined in ISO 230-1:2012. Therefore this location is represented in coordinate system of moving link.

As previously mentioned in this section, the location error is an average value that is to be associated to a kinematic pair independent of mechanism state. Hence, instead of the mechanism state as shown in Figure 4, the mechanism itself should be referred to as used_representation of the item_identified_representation_usage.

3 IMPLEMENTATION

To evaluate the capability of the proposed modelling approach, a prototype application is developed to translate an ASME B5.59 [ASME, 2008] XML machine tool data model to STEP AP242 model. The original data is based on version 13b of the draft standard ASME B5.59-2 (Data Specification for Properties of Machine Tools for Milling and Turning). Performance and capability data of the machining operation is described in the original XML file.

An important aim of the implementation is to perform and evaluate the AP242-based data integration between geometric model and kinematic error model for one machine tool. A major benefit of STEP AP242 is the high level of integration between different aspects of the described product, e.g. geometry, kinematics, classification, tolerances. Multiple levels of details and aspects of the machine tool model can be modelled by different applications and integrated in a single model. An AP242 model can be read and written by different CAD/CAM systems during its lifecycle without the risk of unwanted information loss. It also means functionality of STEP translators in existing systems can be reused for standardised exchange of kinematic error data. Therefore, the data integration needs to be evaluated by an implementation. In this research, the development takes the kinematic error model defined by ASME B5.59-2 and its shape geometry exported from CAD software as an example.

The application is developed underlying a STEP and Java based integration development project, STEP Toolbox. The project aims to provide a maintainable, reusable, extensible, and flexible programming interface to simplify STEP implementation for CAD/CAM developers. Via ISO 10303-22, SDAI (Standard Data Access Interface), STEP p21 file and other supported file types can be manipulated by several kinds of programming language. The STEP Toolbox provides modularised Java programming interface to process different types of product information in engineering-oriented perspective, e.g. geometry, kinematics, classification. Thus, with little STEP knowledge, developers can produce their own applications or plugins for CAD/CAM software based on the toolbox. During this implementation, the kinematic error data

processing for AP242 is integrated within the kinematics module of STEP Toolbox. And the developed ASME/STEP translator performs its functions by using the integrated interface for kinematics within the toolbox.

In order to integrate kinematic error with the existing kinematics module of the STEP Toolbox, an UML (Unified Modelling Language) model is derived from the kinematic error ontology model (Figure 2). In Figure 5, a simplified class diagram for kinematic error is partly displayed underlying design of the STEP Toolbox.

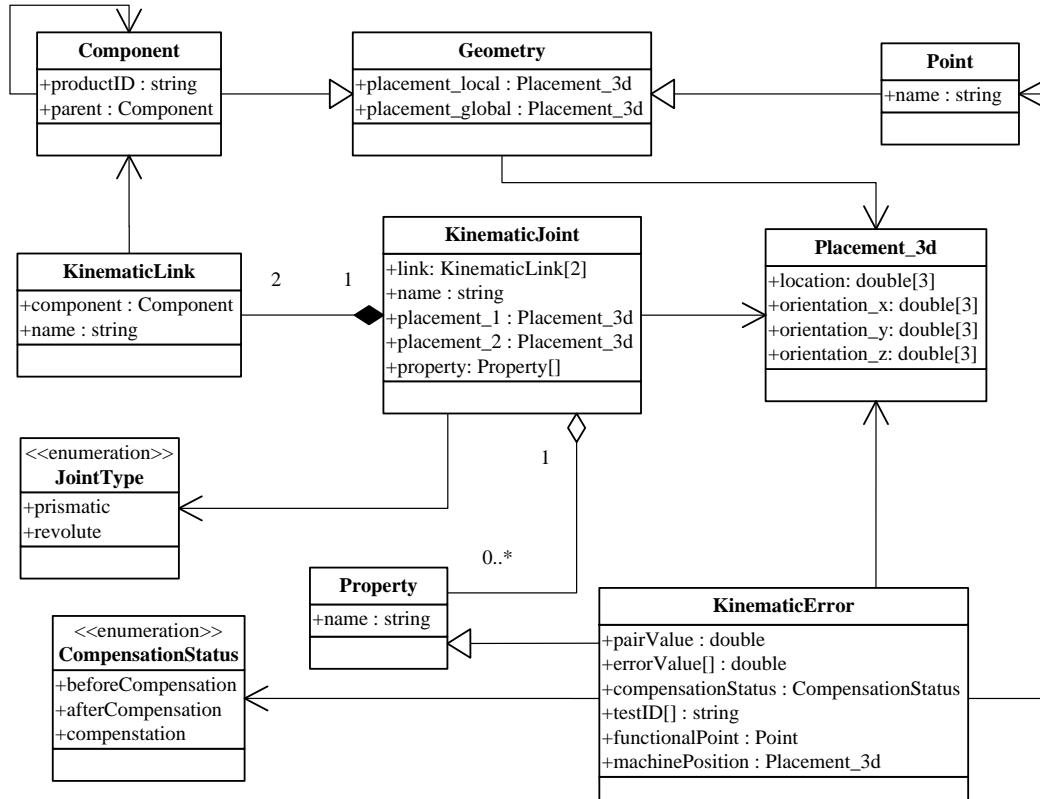


Figure 5: Simplified class diagram for kinematic error

As described in the ontology model, component and point are both subclass of geometry which maintains placement information for general usage. Objects of both links and joints contain both topological and geometrical information. Property is also a general class that is actually used not only by the kinematic joint. Kinematic error is the subclass of the property which is aggregated in the joint. Pair value is a pair parameter for special usage, and is a parameter with a collection of component error. For location error, the developer can simple define a null value for pair value which indicates that the value is independent of the mechanism state.

The system design for the implementation is illustrated in Figure 6. An ASME XML reader is developed to read the component error described in the XML file. The STEP Toolbox is used to retrieve the geometric and kinematic information from the AP242 file. To generate the kinematic error in the STEP model, the kinematics module

in the toolbox is reused and updated and new data model of kinematic error is added for this project, which embodies the maintainability and reusability of the STEP Toolbox. The sample input ASME XML file is provided by NIST within ISO TC184 SC4 WG3 T24 collaboration, which contains machining capability information of a DMG70 machine tool. The STEP file with corresponding geometry and kinematics is produced by the KIBOS for NX (Li et al. 2011). A valid AP242 file is exported with the added pair values, functional points, kinematic error data, and classification as described previously.

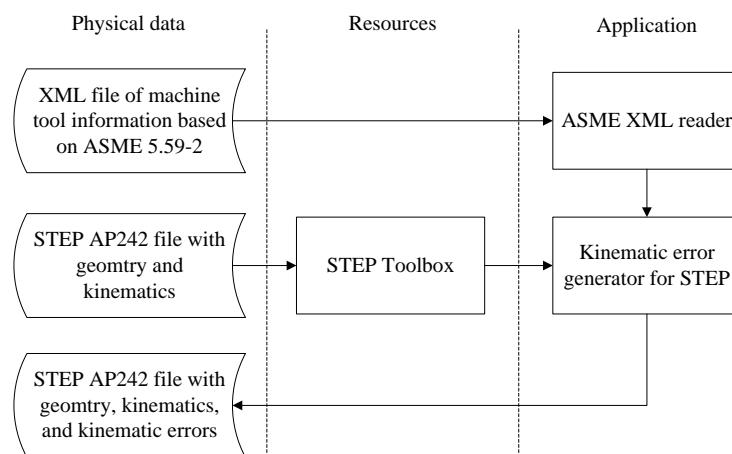


Figure 6; System design for ASME/STEP translator

4 CONCLUSION

This research focuses on the standardised modelling approach for kinematic error. A computer interpretable model for data exchange and integration is proposed and implemented. With the data model and implementation based on STEP AP242, this research utilizes benefits of the representation structure updated in p105 ed2 and demonstrates a modelling methodology applied in practice to reuse the general representation data model in kinematic mechanism. Data representation and exchange between CAD/CAM/CNC systems can be achieved by integration with existing standard translators. Multiple practical functionalities on kinematic error are supported with the proposed data model, e.g. measurement, prediction, and compensation. The solution is promising to be extended to the representation of any other types of properties and classification.

Currently, more data sets, schemas, and measurement practices are being involved in the research. The presented data model is being improved and expanded during discussion with international partners. As a contribution to the ISO TC184 SC4 standardisation work, this modelling approach will be a base for schema level extension in future standard development.

5 ACKNOWLEDGEMENT

We are grateful for the support from VINNOVA and XPRES (Initiative for excellence in production research), and for fruitful discussions with NIST and other members of ISO TC184 SC4 WG3 T24.

REFERENCES

- [**Donmez et al., 1986**] Donmez, M. A.; Blomquist, D. S.; Hocken, R. J.; Liu, C. R.; Barash, M. M.; "A general methodology for machine tool accuracy enhancement by error compensation"; In: *Precision Engineering*, 8(4), 1986, pp. 187-196; ISSN 0141-6359
- [**Altintas et al., 2005**] Altintas, Y.; Brecher, C.; Weck, M.; Witt, S.; "Virtual Machine Tool"; In: *CIRP Annals - Manufacturing Technology*, 54(2), 2005, pp. 651-673; ISSN 0007-8506
- [**ASME, 2008**] ASME B5/TC56; "Part 2, Data Specification for Properties of Machine Tools for Milling and Turning"; In: *Information Technology for Machine Tools*; 2008; ASME B5.59-2, version 13b
- [**Euler-Chelpin, 2008**] Euler-Chelpin, A. von; "Information Modelling for the Manufacturing System Life Cycle"; *Doctoral Thesis*; KTH Royal Institute of Technology, Stockholm, Sweden 2008; ISSN 1650-1888
- [**Hedlind et al., 2010**] Hedlind, M.; Lundgren, M.; Li, Y.; Archenti, A.; Kjellberg, T.; Nicolescu, C. M.; "Kinematic structure representation of products and manufacturing resources"; In: *Proceedings of CIRP 2nd International Conference on Process Machine Interactions*; Vancouver 2010; ISBN 978-0-9866331-0-2
- [**Hedlind et al., 2011**] Hedlind, M.; Klein, L.; Li, Y.; Kjellberg, T.; "Kinematic structure representation of products and manufacturing resources"; In: *Proceedings of CIRP 7th International Conference on Digital Enterprise Technology*, pp. 340-347; Athens 2011; ISBN 978-960-88104-2-6
- [**ISO, 2012**] ISO TC184 SC4; "Managed Model-based 3D Engineering"; Application protocol; Committee draft, ISO/CD 10303-242:2012
- [**Kjellberg et al., 2009**] Kjellberg, T.; Euler-Chelpin, A. von; Hedlind, M.; Lundgren, M.; Sivard, G.; Chen, D.; "The machine tool model—A core part of the digital factory"; In: *CIRP Annals - Manufacturing Technology*, 58(1), 2009, pp. 425-428; DOI 10.1016/j.cirp.2009.03.035
- [**Li et al., 2011**] Li, Y.; Hedlind, M.; Kjellberg, T.; "Implementation of kinematic mechanism data exchange based on STEP"; In: *Proceedings of CIRP 7th International Conference on Digital Enterprise Technology*, pp. 152-159; Athens 2011; ISBN 978-960-88104-2-6
- [**Schwenke et al., 2008**] Schwenke, H.; Knapp, W.; Haitjema, H.; Weckenmann, A.; Schmitt, R.; Delbressine, F.; "Geometric error measurement and compensation of machines – An update"; In: *CIRP Annals - Manufacturing Technology*, 57(2), 2008, pp. 660-675; ISSN 0007-8506

Paper B

Li, Y., Hedlind, M., Kjellberg, T. and Sivard, G. (2013a). Cutting tool data representation and implementation based on STEP AP242, *Smart Production Engineering*, pp. 483-492, Berlin/Heidelberg: Springer-Verlag. DOI: 10.1007/978-3-642-30817-8_47.

Cutting Tool Data Representation and Implementation Based on STEP AP242

Yujiang Li¹, Mikael Hedlind¹, Torsten Kjellberg¹, Gunilla Sivard¹

¹ Production Engineering, KTH Royal Institute of Technology, Stockholm, 10044, Sweden

{yujiang.li, mikael.hedlind, torsten.kjellberg,
gunilla.sivard}@iip.kth.se

Abstract. For cutting tool data exchange in manufacturing CAx (Computer-Aided technologies), standardized representation and classification of items and properties is important. ISO 13399 (Cutting tool data representation and exchange) provides a solution to represent cutting tool data classified with an ISO 13584 (Parts Library, PLib) based dictionary. However, ISO 13399 does not support classification of shape geometry directly, which limits its use. Another limitation is representing GD&T (Geometric Dimensioning and Tolerancing) as simplified general properties, which does not fulfill high semantic precision and validation rules. This research provides a unified solution to represent cutting tool parameters integrated with geometry and dedicated properties based on STEP AP242 (ISO 10303-242 Managed model-based 3D engineering). Standardized libraries such as the ISO 13399 dictionary can be reused with the modeling approach for AP242 cutting tool representation. Software is developed to validate and demonstrate how this solution facilitates the data integration process to support CAx applications.

Keywords: Modeling, STEP AP242, Cutting tool, Classification

1 Introduction

Collaborations between enterprises put high requirements on seamless digital information exchange and sharing between software systems for industry. Standardized representation and classification of items and properties is fundamental for cutting tool data exchange between applications of PLM, CAD/CAM and CNC (Product Lifecycle Management, Computer-Aided Design, Computer-Aided Manufacturing, and Computer Numerical Control). With PLM applications, engineering information of cutting tools is integrated to support its development and deployment through the lifecycle. With CAD applications, cutting tool geometry, assembly structure, and relevant properties are defined in different viewpoints for production and utilization. With CAM applications, cutting tool requirements and usages are defined for operations with tool paths and cutting parameters. With CNC applications, operations are executed with input on cutting tool types, main dimensions, and tolerances on tool wear. For decades, information management has been regarded as the essence of cut-

ting tool management for efficient computerized manufacturing control [1]. However, differences in terminology and data format between different computer-aided software are blocking cross-system interoperability.

An unambiguous way for the cutting tool data modeling and exchange is ontology based on industrial standards. A standardized hierarchy of classes and properties is the basic structure to describe taxonomies, as for cutting tool ontology. An important contribution in this area is the dictionary of cutting tool classes and property types within ISO 13399 (Cutting tool data representation and exchange). The dictionary is standardized underlying PLib (ISO 13584, Parts Library). A data set conforming to ISO 13399 is cutting tool data representation including its classification referencing the dictionary. On the other hand, shape representation is set outside the scope of ISO 13399. If needed, geometric models in other formats, e.g. STEP (ISO 10303, STandard for the Exchange of Product data), can be referenced from the data set. This separation approach is traditional for PLM systems.

A barrier for implementation based on ISO 13399 is the multiple data sets and associated data schemas that must be managed. For instance, an ISO 13399 cutting tool data set is transferred from a tool supplier to its customer with a corresponding geometry data set of STEP AP214 (ISO 10303-214, Application protocol: Core data for automotive mechanical design processes). System developers have to establish and maintain data integration based on the two standards and find a convenient way to convey both data sets without data loss. ISO 13399-1 is mainly a subset of AP214, which makes the practical situation even worse. As a result, there are two standards with mainly equivalent schemas to be synchronized for implementation and maintenance. Besides, the ISO 13399-150 usage guidelines describe implementation differently from how it is done for STEP AP214. It results in great efforts for implementers to develop and coordinate separate implementations.

The design of schemas also makes ISO 13399 differ from STEP in representation structure. It can be observed that both standards share the same functionality of GD&T (Geometric Dimensioning and Tolerancing) representation but with different modeling approaches. In STEP, the dedicated representation schema for GD&T is a basic functionality that is preferably reused for cutting tools, and the connection to geometry can be easily established. ISO 13399 instead uses general properties without a geometric context for shape dimensions. This does not meet high requirements for data consistency and interpretation precision in future CADCAM and CNC applications. A lack of a common schema for the geometry and GD&T results in a lack of a complete context and an unambiguous representation. With the representation of e.g. diameter classified as cutting diameter, the link to a specific geometry shape element is important information required in industry.

A standardized approach to represent cutting tool information is important for modern industry. The dictionary provided by ISO 13399 is promising for a system independent cutting tool library of CAM systems, which is able to establish interfaces between various tool makers and their customers [2]. Since it fulfills its defined scope for cutting tool data representation, ISO 13399 has contributed to several researches. Kaymakci et al. [3] adopt concepts from ISO 13399 for a general prediction model of inserted cutters, where the demands for “a unified geometric, kinematic, and mechan-

ics model” are not what ISO 13399 is able to meet solely. Helgason and Kalhor [4] use ISO 13399 for cutting tool data exchange in the context of machining process planning, where the solid model and cutting tool parameters are separately represented with STEP format and ISO 13399. Chungoora et al. [5] highlight the problems for joints usage of standard and present an ontology-based framework to consolidate various production information standards, e.g. ISO 10303, ISO 13399, ISO 13584, and ISO 15531. Generally speaking, integrated geometry models are commonly required regarding the adoption of ISO 13399. It becomes a must that multiple standards are integrated to achieve a complete modeling solution. Therefore, a modeling approach with a unified standard architecture such as STEP will be helpful for cutting tool data exchange.

Within the framework of STEP AP214, dimensions and tolerances are defined in a specific UoF (unit of functionality). Classification of items based on PLib is supported by AP214 in another UoF, which provides association with any dictionary conforming to PLib, e.g. the ISO 13399 cutting tool library. The draft standard AP242 (ISO 10303-242 Managed model-based 3D engineering) is going to replace AP214. Added capabilities in AP242 include supports for the Geometrical Product Specifications (GPS) standard. Dedicated schemas for GD&T and external references are integrated into this protocol and can be used to represent the needed information for CAD/CAM and CNC cutting tool data exchange as well as PLM applications.

In this research, STEP AP242 and ISO 13399 cutting tool library are combined for the comprehensive cutting tool modeling. Products, GD&T, features, general properties, and classes are basic elements in the model. The following section introduces the standardized modeling approach based on AP242, and the mapping strategy to a UML data model for implementation. The third section presents a prototype implementation which is able to classify AP242 cutting tool data sets with the ISO 13399 PLib-based dictionary.

2 Cutting Tool Modeling

Multiple types of information regarding cutting tool data are considered in this research, which requires data integration among schemas. The standardized product generic information modeling schema in STEP is able to support association of classes and properties defined in the ISO 13399 dictionary with products, shape elements, GD&T, features, and other elements. This capability also applies for different levels of details for different user requirements e.g. geometric level of details. For instance, tool suppliers use a complete model with a high level of detail for internal data exchange, but tool customers may only need the basic cutting tool parameters. Thus, exact geometry and other detailed design requirements, which in many cases also are confidential or unnecessary, should be trimmed from the complete model for external data exchange.

Figure 1 presents a STEP AP242 data excerpt of a diameter representation associated with a face on a solid body representation. In the model, a diameter of 22.0 mm with a tolerance of 0.05 mm and -0.1 mm is associated with a face. All geometric

information is represented in the established way for STEP, which is omitted in the empty block. Using the standard modeling concept of shape aspect, dimension definition, measure representation and other properties are precisely integrated.

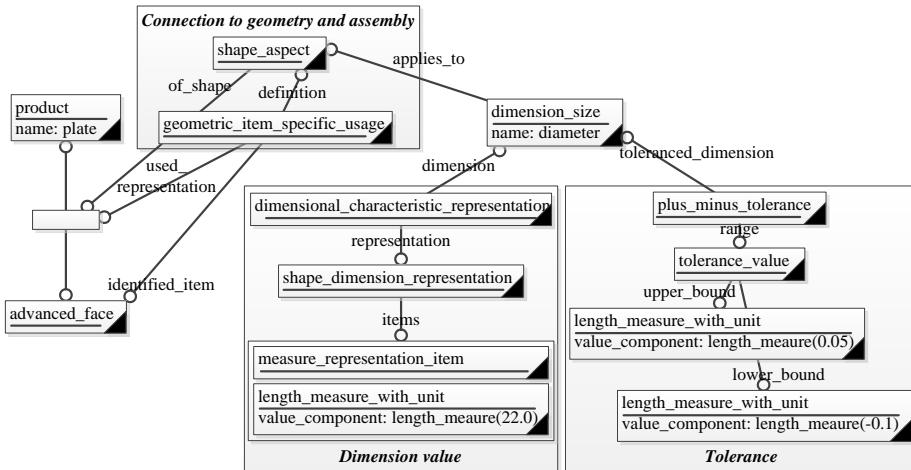


Fig. 1. Example of geometric dimension modeling in STEP AP242

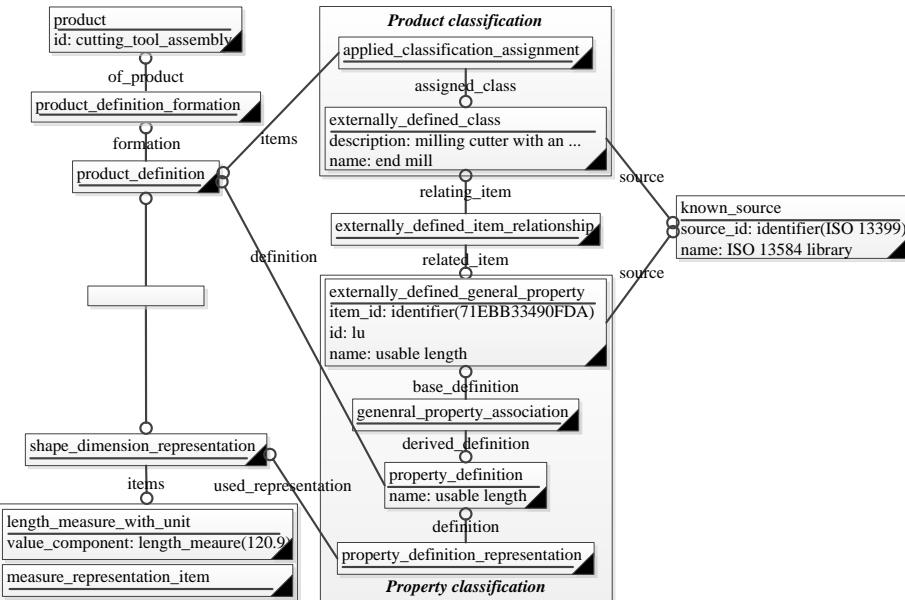


Fig. 2. Example of classification modeling in STEP AP242

Business cases may require hidden shape information, and then there will be no link between the dimension definition and the face. Nevertheless, the dimension itself is represented in a fixed way independent of existence of shape information. For a complete definition of GD&T, a geometric context is needed to establish coordinate system for the dimensions, e.g. supplemental geometry. Supplemental geometry is also known as help geometry or constructive geometry. Example data types are placements, points, curves and faces. These data types are not used to define shapes, but to support the definition of design requirements. It is a common CAD functionality also available in STEP. Recommended practices for implementing supplemental geometry are published by the CAx Implementers forum [6].

Figure 2 exemplifies the classification modeling approach within AP242. The ISO 13399 cutting tool dictionary is referenced in the model. The description of an end mill with a usable length is retrieved from the library and associated with the corresponding product and dimension.

3 Implementation

As an information modeling approach for computational applications, the result of this research should be validated with software implementation. The presented prototype software aims to evaluate the feasibility of the proposed solution and to provide a development strategy for industrial applications in the future.

The major function of the software is to classify products and other properties based on AP242 referencing the ISO 13399 cutting tool library based on PLib. The input is a p21 (ISO 10303-21, clear text encoding of the exchange structure) file of AP242 with its assembly structure, shape features, dimensions with tolerances, and other properties. Shape representation or supplementary geometry is optional. In a case study, the input STEP file is exported from a plug-in to Siemens NX 8.0 that is developed in this research project, of which the integration strategy is demonstrated in [7]. Assembly tree of cutting tool with the identification of each occurrence is displayed as a product part list breakdown. Thus, product parts, shape features, GD&Ts, and general properties can be classified according to PLib in an unambiguous way.

The development of this software reuses and contributes to a Java-based development project, STEP Toolbox. The scope of the STEP Toolbox project has a larger scope than product classification, e.g. kinematics and geometric errors [8]. In general, this project aims to significantly improve the usability of STEP standard for its major readers, CAx system developers. A complete solution to simplify STEP implementation is presented as a programming interface with high maintainability, reusability, and extensibility. The toolbox provides modularized functions to process different types of product information in engineering oriented perspective, i.e. to manage integrated geometry, kinematics, classification and other kinds of product data. Without requiring STEP knowledge, developers can produce their own standalone applications or plugins for CAD/CAM software based on the toolbox. The following data model design is a result of the research presented in this paper as well as a contribution of STEP Toolbox.

3.1 Data Model

STEP Toolbox aims to provide a friendly programming interface for CAx developers, rather than designers in a specific industrial domain. Therefore, concepts should be defined in a domain-independent way so that elements are reusable in multiple programming modules in the toolbox, e.g. the component is a widely used concept in geometry, kinematics, and GD&T. Thus, the toolbox solution adopts ontology model mapping from EXPRESS (ISO 10303-11) schemas in order to generate a proper design of computer interpretable UML model for programming. The ontology also helps advanced programmers to understand the conceptual relationship between EXPRESS models and UML models. Using the modeling principle outlined by Kjellberg et al. [9], an ontology model for cutting tool implementation has been used for mapping from AP242 in EXPRESS to a UML model (see Figure 3).

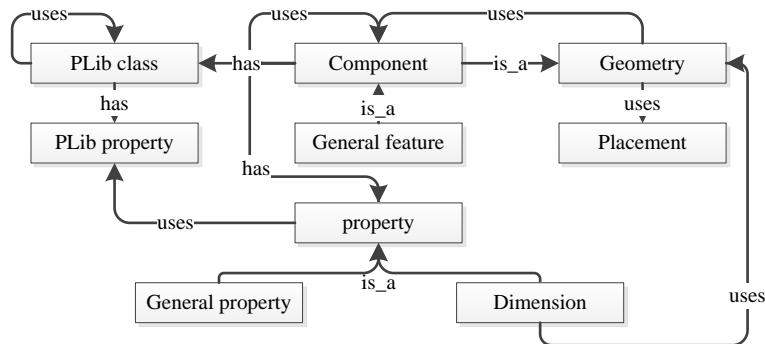


Fig. 3. Cutting tool implementation ontology

Note that the presented ontology model mainly serves the purpose of supporting computational implementation going from EXPRESS to UML. It does not necessarily express semantic relationship precisely. For example, a component actually does not own PLib class, but it can be multiply defined by several classes, which leads to such a composition relationship, e.g. a multi-functional cutting tool can be classified as an end mill and a drill. Another example is that semantically the general feature should not be defined as a sub-class of the component. However, it also can be classified and has classifiable properties, which indicates all the requirements of the general feature implementation can be met by the attributes and operations of the component. As a result, the general feature is set as the subclass of the component here. Geometry is a general concept, of which the subclasses include bodies, faces, edges, and points, besides the displayed components in the figure. Both the PLib class and the component have self-references to indicate the tree structure for presentation. Properties such as general properties and dimensions are associated with certain components or general features, which are defined semantically by PLib Property, e.g. a linear distance associated with a component is defined as usable length in a PLib dictionary (see Figure 2). PLib properties have strict ownership from certain PLib classes, which

can only classify the properties of the components classified with corresponding owner PLib classes.

Programmers can use the ontology model to understand concepts easily, but a well-designed UML (Unified Modelling Language) model is fundamental for practical implementation. As a core part of UML, a class diagram is important of such a Java-based programming interface. In Figure 4, a simplified class diagram for cutting tool development is illustrated. Methods of most classes are not displayed, but they are important for implementation, such as getters, setters, and constructors. As a part of STEP Toolbox, modules are divided and controlled by managers, such as the property manager and the classification manager. The STEP model manager provides basic operations for STEP data set, such as initialize, export, and close. Differences from ontology may occur for specific data model implementation. For example, both components and PLib classes have tree structures, but PLib classes also need to record a list of parents, rather than only the direct parent which is the case for component. This is caused by the inheritance of PLib properties from all relative parents.

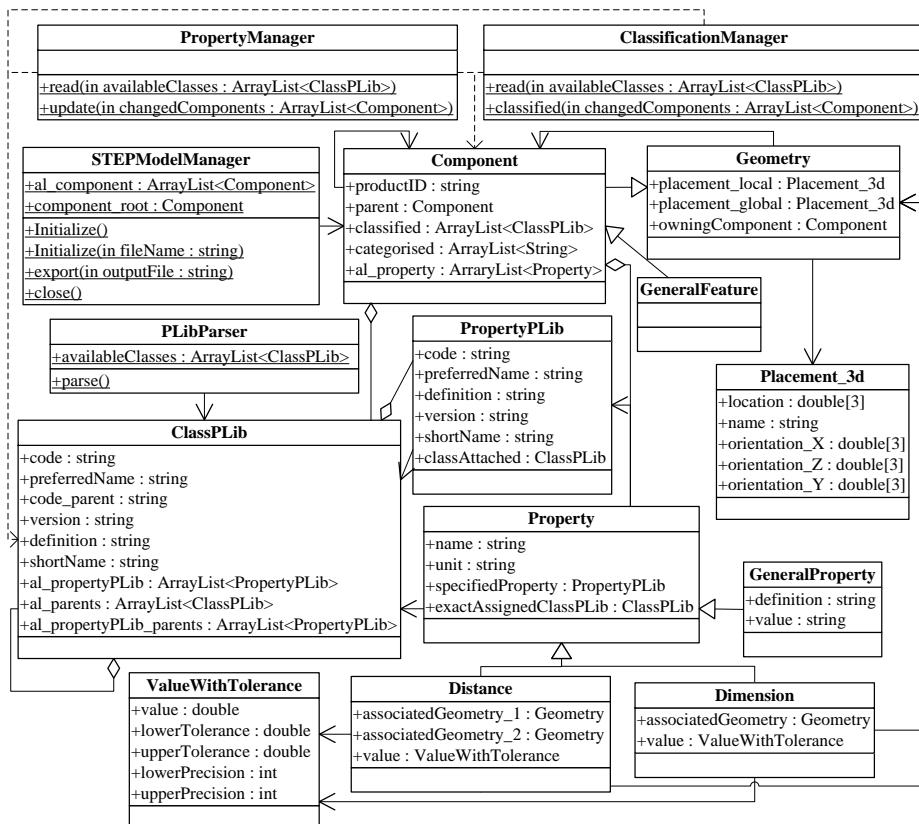


Fig. 4. Simplified class diagram for cutting tool development

3.2 System Development

An MVC (Model-View-Controller) structure design is illustrated in the Figure 5. The relative part of STEP Toolbox is integrated as the Model layer. The system starts with a file opening dialog. If the file exists and is acceptable, the Initializer triggers the STEP model manager to read the file and invokes the PLib parser to generate the standard definitions of classes and properties. The GD&T manager and the classification manager need to analyse the STEP model, read dimensions, general properties, and classification information. Then the assembly controller collects all the data and invokes the assembly viewer to display the assembly with GD&T in the graphical user interface (e.g. see Figure 6). The operation of classification is started by a selected component from assembly. A classification dialog is used to organise the PLib classes and properties in a displayable and selectable way (e.g. see Figure 7). The classifier in the controller classifies the products and properties only in the level of Java data model rather than the STEP model, and records the updated component whenever there is a user operation. Then, the exporter triggered by interface performs all the changes to the data set and exports it with specific configurations.

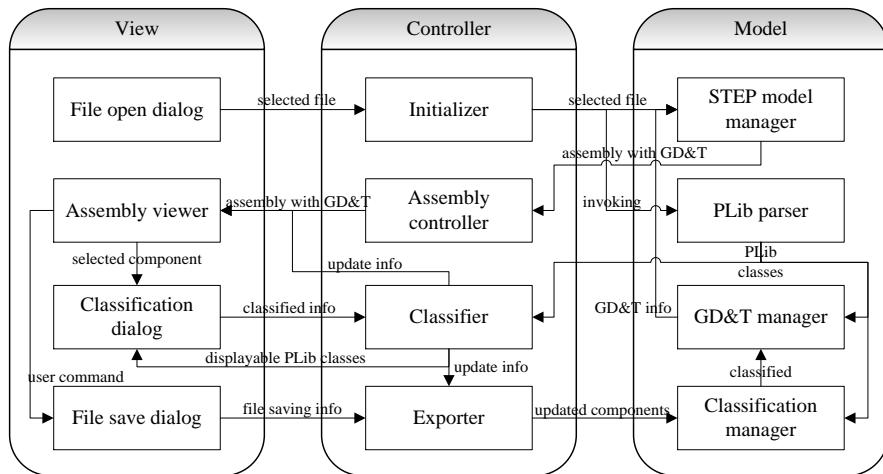


Fig. 5. MVC system design

3.3 Interface Design

The interface is designed to present and manage different elements in the data set with a clear structure. Figure 6 illustrates the basic design of the main user interface. Components are presented in a tree structure as commonly used in CAx systems. Features are displayed with a darker background color and a special category name. Both components and features can be selected to classify. The bottom table displays related properties of the selected item in the above tree table.

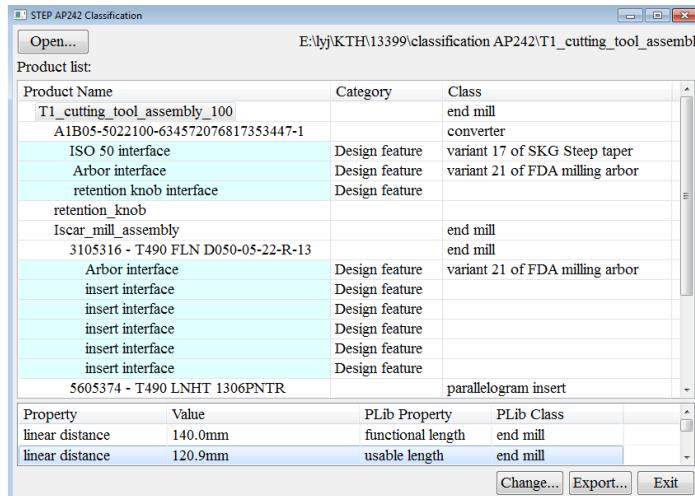


Fig. 6. Main user interface

The major function of this software is performed after the click of the “Change...” button. Then the classification dialog (see Figure 7) pops up with two lists of selection: a tree list of PLib classes that can be multiply selected, and a table list of PLib properties of the selected class. The combo list contains all available properties owned by the selected component/feature to be specified, e.g. the linear distance is specified as the function length of an end mill in Figure 7.

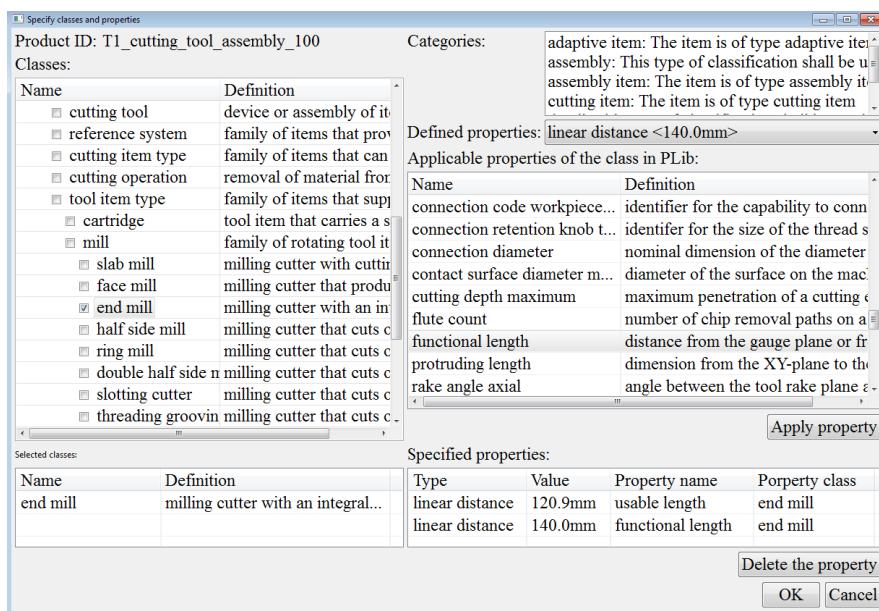


Fig. 7. Classification dialog

4 Conclusion

This research focuses on the description and evaluation of a standardized cutting tool data modeling approach. A unified computer interpretable model for data exchange and data integration is proposed and implemented. Benefits of the comprehensive representation structure of STEP AP242 are utilized and demonstrated in developed application. External standardized library, such as the ISO 13399 PLib-based dictionary, can be easily reused and associated with the shape representation. Practical data exchange is promising by integration developments with current commercial CAx systems. With the unified model, industrial practitioners can skip the barrier to coordinate multiple standards and achieve integration development smoothly.

Acknowledgments. We are grateful for the support from VINNOVA and XPRES (Initiative for excellence in production research), and for fruitful discussions with members of ISO TC184 SC4 WG3 T24 and industrial practitioners from Sandvik Coromant.

References

1. Veeramani, D., Upton, D.M., Barash, M.M.: Cutting-Tool Management in Computer-Integrated Manufacturing. International Journal of Flexible Manufacturing Systems. Vol. 4, no. 3/4, 237-265 (1992)
2. Zelinski, P.: Coming Soon: Universal, CAM-Independent Cutting Tool Library. Modern Machine Shop. Vol. 84, no. 4, 22-24 (2011)
3. Kaymakci M., Kilic Z.M., Altintas Y.: Unified Cutting Force Model for Turning, Boring, Drilling and Milling Operations. International Journal of Machine Tools and Manufacture. Vol. 54-55, 34-45 (2012)
4. Helgason M., Kalhorri V.: A Conceptual Model for Knowledge Integration in Process Planning. Procedia CIRP. Vol. 3, 573-578 (2012)
5. Chungoora N., Cutting-Decelle A.-F., Young R.I.M., Gunendran G., Usman Z., Harding J.A., Case K.: Towards the ontology-based consolidation of production-centric standards. International Journal of Production Research. Vol. 0, 1-19 (2011)
6. Bay, J., Rosché P.: Recommended Practices for Supplemental Geometry, Release 1.0. CAx Implementor Forum (2010)
7. Li, Y., Hedlind, M., Kjellberg, T.: Implementation of Kinematic Mechanism Data Exchange Based on STEP. In: 7th CIRP-Sponsored International Conference on Digital Enterprise Technology, pp. 152-159. Athens (2011)
8. Li, Y., Hedlind, M., Kjellberg, T.: Kinematic Error modeling Based on STEP AP242. In: 1st CIRP Sponsored Conference on Virtual Machining Process Technology. Montreal (2012)
9. Kjellberg, T., Euler-Chelpin, A.V., Hedlind, M., Lundgren, M., Sivard, G., Chen, D.: The Machine Tool Model — A Core Part of the Digital Factory. CIRP Annals - Manufacturing Technology. Vol. 58, no. 1, 425-428 (2009)

Paper C

Li, Y., Huang, Q., Hedlind, M., Sivard, G., Lundgren, M. and Kjellberg, T. (2014a). Representation and exchange of digital catalogues of cutting tools. *Proceedings of ASME 2014 International Manufacturing Science and Engineering Conference*, Detroit. DOI: 10.1115/MSEC2014-4131.

MSEC2014-4131

REPRESENTATION AND EXCHANGE OF DIGITAL CATALOGUES OF CUTTING TOOLS

Yujiang Li

KTH Royal Institute of Technology
Stockholm, Sweden

Qiuling Huang

KTH Royal Institute of Technology
Stockholm, Sweden

Mikael Hedlind

Scania CV AB
Södertälje, Sweden

Gunilla Sivard

KTH Royal Institute of Technology
Stockholm, Sweden

Magnus Lundgren

KTH Royal Institute of Technology
Stockholm, Sweden

Torsten Kjellberg

KTH Royal Institute of Technology
Stockholm, Sweden

KEYWORDS

Information Modeling, Digital Catalogue, ISO 10303 STEP,
ISO 13399, ISO 13584 PLib

ABSTRACT

Information management for manufacturing resources such as cutting tools is an important research topic in the context of cloud manufacturing. Vendors and customers usually use catalogues to communicate information for such manufacturing resource. Incompatibilities of information in syntax, semantics, and structure among supply chains often result in inefficient manual sharing and management of the catalogue information. It is difficult for cloud based applications to pool information from various sources. This communication failure calls for a system neutral solution for data modeling and exchange to enhance interoperability of the cutting tool catalogue information. Previous studies have present solutions for representation of the cutting tool information with STEP AP242 (ISO/DIS 10303-242) with semantic classification referring to a PLib (ISO 13584, Part Library) based dictionary. This approach can be extended for the catalogue modeling, due to functionalities for specification and configuration control of general product variants in the same standard.

With a modeling approach with standardized information schemas, system architecture to guide implementation is proposed to enhance the communication in practice. Relative elements to represent vendors' catalogues and customers' requirements are modeled. Associations to the PLib-based dictionary complete semantics and enable information mapping between vendors and customers. Principles of the mapping are identified to facilitate implementation of related software

systems. Prototypes are developed to verify the proposed system architecture. The proposed solution is promising to migrate to other types of products than cutting tools, because the data models are based on the general product models defined in AP242.

INTRODUCTION

One reason of digitalization in every aspect of manufacturing industry is a need to improve communication in supply chains. A major challenge is enhanced interoperability in digital systems for design, planning, production, marketing, etc. Standardization is an important option to harmonize syntax, semantics and structure of information.

Information management of manufacturing resources and related user requirements is a key challenge for cloud manufacturing. Cutting tools are one of the fundamental resources for modern production systems. An efficient way to manage digital information is important through the life cycle of cutting tools, from CAD for product design and CAM for process planning to CNC controller for production execution [1]. How to lower management effort of such resource information is a key issue in an environment of service-oriented cloud manufacturing.

Problem description

Paper or web browser based catalogues of cutting tool information are common approaches for vendors to communicate technical data to customers. Vendors commonly use general PLM systems to manage categorization of their product data in catalogs. Native modeling interfaces of the PLM software may be utilized to facilitate semantic information about the products and properties. Meanwhile, their customers usually do not have the same software to manage the

purchased resources. Moreover, it is observed that usually preferences of categorization are different between two sides of procurement. Therefore, schematic incompatibilities in semantics, structure, and syntax are a barrier for efficient digital communication between the vendors and the customers.

Information modeling of cutting tools

Formal information models are an important solution to represent facts, concepts, or instructions with specified requirements (ISO 10303-1). Standardized information modeling can serve as a carrier for centralized management and encapsulated resources which are preferred to implement cloud manufacturing [2] in a process to virtualize and package services. For product information management in the cutting tool industry, ISO 13399 (Cutting tool data representation and exchange) introduces modeling methodology using a dictionary compliant to ISO 13584 (Parts Library, PLib [3]). Semantic ontology for products and properties is standardized in this solution [4]. However, cutting tool catalogue representation is not in the scope of ISO 13399.

The functionality introduced in STEP AP214 (ISO 10303-214) for specification and configuration control is developed to describe product variants in general. The usage of this standard was analyzed and extended by Sivard [5] to deliver a Generic Information Platform for product family. The same functionality has also been introduced in the latest version of STEP AP203 (ISO 10303-203) of which product concepts, classes, specifications, and configurations are supported in a group of conformance options. These solutions are unified and adopted during the development of STEP AP242 (ISO/DIS 10303-242) which becomes a promising protocol to fulfill the requirements for an efficient solution of cutting tool catalogue communication.

Research objective

The objective of this research is to enable effective and efficient communication for cutting tool catalogues between vendors and customers. Based on a standardized platform, vendors are supported to define product families for their catalogues. Readers are able to view catalogues from different vendors in unified structure that can be customized. Utilization of ISO standards can bridge the vendors and customers with a same context i.e. the vendors can standardize description of their product families and the customers can state and archive their requirements or queries to get desired products in a standardized form. Presentation and comparison of products from different vendors are supported by the proposed solution. Information types of in traditional catalogues may be supported, such as version control, release management, products, product families, and assortments. Other types of information may be easily integrated within STEP AP242 models as options, such as geometry and kinematics. The proposed solution is also applicable for catalogues of any types of products in the manufacturing industry. Contributions of this research include:

- Modeling guidelines and samples based on STEP AP242, PLib, and the cutting tool dictionary of ISO 13399;
- Fundamental principles of creation and utilization of standardized product catalogues;
- And implementation guidelines of software systems to create and use catalogues.

PREVIOUS WORK FOR CATALOGUE MODELING

Support of data exchange for electronic catalogue information is an important reason why PLib was initiated in the 90's. Pierra [6] introduced the early efforts for this standardization work. He pointed out that the goal of the electronic catalogues is transmission of component knowledge from suppliers to developers of computer-aided engineering tasks, e.g. selection and evaluation of components. Interpretabilities for both computer and human are fundamental requirements for the exchange of the electronic catalogues. Modeling approaches for data structure, descriptive knowledge, and procedure knowledge are proposed as a basic mechanism of the PLib-based electronic catalogues. Business models are developed to support supply and usage of the electronic catalogues.

PLib p42 (ISO 13584-42) specifies methodology for structuring parts families. A standardized hierarchy to characterize product by properties with values is introduced to establish representation of product catalogue information. Users of this standard are expected to define a characterization hierarchy or to refer to a characterization class in other standardized product models. However, data models provided in this standard are not advised to be directly used. A number of super entities are defined for further extensions. Therefore, the standard only aims at a solid conceptual basis to model product families.

Efficient E-business calls for a standardization solution for categorization of product information. Albrecht et al. [7] made a summary for marketplace and technology standards to enable communication between buyers and sellers. Three levels of standardization are roughly grouped to support global e-business. The lowest level, serving as building blocks, includes data type standards, expression languages, and communication methods. ISO 10303 STEP can provide a complete solution for this level. Product catalogues are grouped into the second level, marketplace standards. Semantic information is standardized as schemes such as the North American Industry Classification (NAICS) and the United Nations Standard Products and Services Code (UNSPSC). Coded taxonomy with hierarchy is documented as references for E-business. To support interface for human-computer communication, applications and services are developed as the third level.

In the context of electronic components, Bellatreche et al. [8] presented a solution for automatic integration of electronic catalogues with ontology-based database. Data integration of different sources with local schemas is considered as an important challenge for many information management systems. Semantic and schematic heterogeneities are blocking

establishment of a uniform interface for end users' queries. Diversities of local ontologies are expected to be solved by the integration approach with a shared ontology. The modeling with PLib is also explored and utilized in the solution.

Standardization of catalogues in a schematic level is also an issue in Architecture/Engineering/Construction and Facility Management (A/E/C/FM). Nyambayo et al. [9] made a review of relevant techniques. EXPRESS- and XML-based standards are important ways to represent data sets and external reference libraries. Information required in such systems includes product identification, order information, finance, transaction, and delivery, besides classification for semantics. For A/E/C/FM, rapid development of online system to support generation, classification, and exchange of manufactured products inspires many projects for electronic catalogue standardization during the last decade. Armor et al. [10] made a survey in the domain. Unification of data representation and mapping between different aspects of reality in the form of ontology is still concluded as a major barrier to reach a final solution, which is a challenge of this research as well.

A recent conceptual contribution for cutting tool catalogue is GTC (Generic Tool Classification) by Sandvik Coromant. The PLib dictionary of ISO 13399 is used in GTC to describe semantics of cutting tool components which form a standardized generic classification hierarchy. The GTC is supported and integrated by Siemens in a demonstrator of Manufacturing Resource Library (MRL) [12]. The non-leaf levels of the GTC hierarchy are expected to be standardized [11], and vendors may specify their product families into the leaf level of the GTC hierarchy. There are advantages with such

kind of generic ontology for product catalogues, e.g., applications are easy to develop and integrate with existing CAx systems, and all stakeholders can understand the modeling principle relatively fast. However, the GTC approach has limited flexibility to respond to new technologies and new product concepts as a result of vendors' innovation. The maintenance of only one generic hierarchy for the cutting tool catalogue may limit the scope of the modeling result, and block possible extensions to other types of products.

METHODOLOGY

A fundamental challenge of this research is unification and mapping of hierarchies defined respectively by stakeholders of the cutting tool supply chain (Fig. 1). Three groups are concerned in the cloud manufacturing: users, application providers, and physical resource providers [13]. Regarding cutting tool catalogue information in this research, the most important players are users as consumers (who can be generalized as readers) and physical resource providers as vendors who are creators of local hierarchies.

The unification is realized by standardized modeling principles for the local hierarchies with external classifications based on STEP AP242. The cutting tool dictionary defined in ISO 13399 is used as a global ontology that bridges the local hierarchies. It is standardized with PLib p25 (ISO 13584-25, Parts library -- Part 25: Logical resource: Logical model of supplier library with aggregate values and explicit content). The dictionary provides unified semantic information for classification of cutting tool products, features, properties, etc., so that vendors and customers can share a same context.

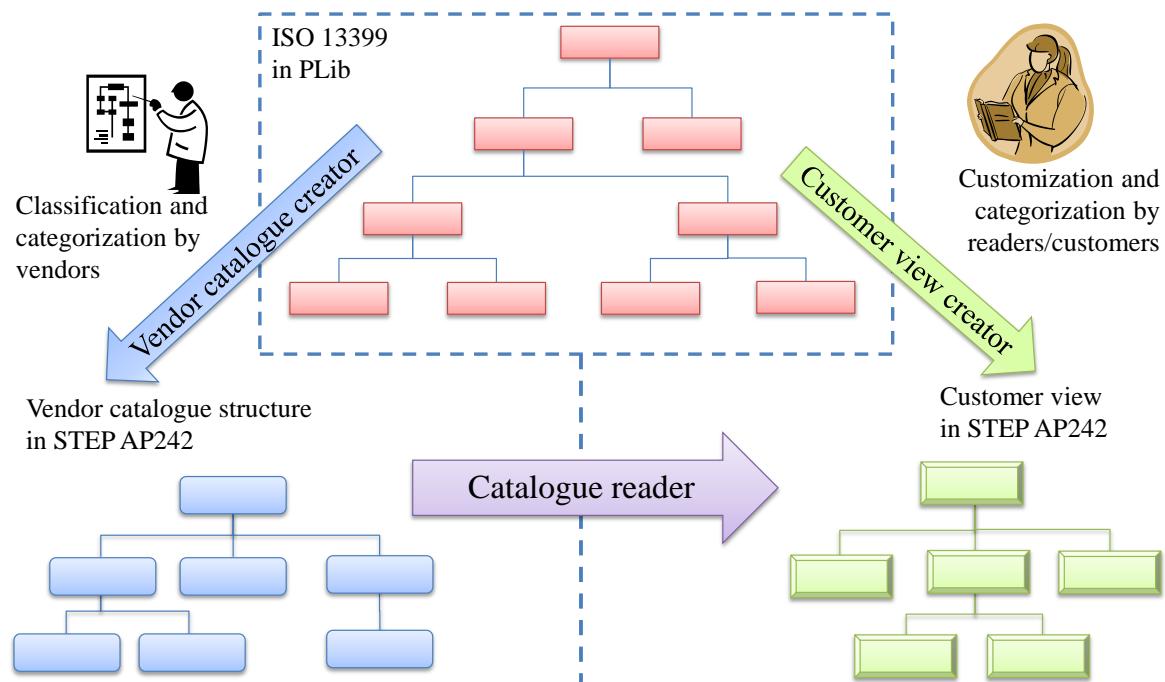


Figure 1. System architecture

One type of the local hierarchies is product catalogues. In this research, standardized information models of digital product catalogues are replacement of the traditional catalogues of cutting tools. Referring to information of design and manufactured products, the digital catalogues can be used to support marketing and customer relationship maintenance. In such information models, product variants are categorized into product classes as nodes in a tree hierarchy with single inheritance. Properties with values or value ranges act as attributes of the nodes optionally. The product classes and properties can be classified by semantic units in the global ontology, the ISO 13399 cutting tool dictionary. For instance, a product class is classified as *end mill*, and the vendor may specify the *cutting diameter* as a certain value or value range to help customers locate the required products. Detailed information of each product class on geometry, interface, properties, functions, etc. can be exported from the CAD systems of vendors, and be imported into the CAM/CAPP/CNC systems of customers.

The other type of the local hierarchies is to represent user requirements. Multiple types of requirements grouped in a hierarchy are usually preferred for catalogue navigation by the customers. Therefore, customer queries are also represented in tree structure which is named as customer view in this research. Modeling principles based on STEP AP242 incorporating a schema for set theory (ISO/TS 10303-1210) are defined for the customer view in this solution. Each node of the customer view hierarchy is a set theory expression with operators e.g. union, intersection, and complement. Standardized classes (optionally with properties) from the cutting tool dictionary are referred to as operands in the set theory expressions. Values or value ranges of properties should be specified for the operands with properties. The customer view acts as presentation guidelines of the catalogues for customers. In the product catalogues based on STEP AP242, each product class is treated as an element of a set defined by its classification. Thus, dedicated software systems can map the product classes from the vendor catalogue to the node of the customer view according to the classification of the product classes.

In this way, the cutting tool dictionary plays a role as the global ontology bridging semantics in the local hierarchies defined by the vendors and the customers. Data models based on STEP AP242 should support information modeling and communication not only for the vendor catalogues, but also for the customer view (see section Modeling Approach).

Mapping between the hierarchies and reality are important for successful implementations. Principles should be defined for mapping between vendors' products and the vendor catalogues, mapping between customers' requirements and the customer view hierarchy, and mapping between the local hierarchies and the global ontology. This is the reason why three types of implementations (see the three arrows in Figure 1) are designed in the system architecture of this solution. A catalogue creator should be used by the vendors to create the vendor catalogues according to their product families, marketing strategies, and the cutting tool dictionary. A customer

view creator is used by the related users to configure the customer view ontology according to their requirements of procurement and utilization. A catalogue reader can present the catalogue according to not only the hierarchy defined by the vendors, but also the hierarchy defined as in the customer view. The three types of implementations propose three basic functions that can be executed by same or different users. They are not necessarily isolated in real applications. For instance, the customer view creator and the catalogue reader can be provided as a merged software package to the customers, since the target user groups of both applications are overlapping to some extent.

Regarding procedures, constraints, and recommendations, principles on creation and maintenance of the local hierarchies should be integrated to the implementations to enable convenient communication between the two sides (see section Mapping Principles).

MODELING APPROACH

Convenient data exchange should be supported by standardization of information modeling. This section presents modeling approaches for the vendor catalogues and the customer view based on STEP AP242, i.e. the two local hierarchies in Figure 1. Instantiated data excerpts in the form of AP242 MIM (Module Integrated Models) are used to exemplify the proposed information models. All proposed data models have been created by prototype implementations and validated against corresponding schemas.

Modeling of the vendor catalogue hierarchies is a fundamental result of this research. In one of the most common STEP application protocol, AP214, a UoF (Unit of Functionality), specification control, provides capability to describe product families in general. Basic functionalities include identifications of similar products, characterization of products with specifications, decomposition of product structure, physical instances of manufactured parts, etc. This UoF is included in the latest developed application protocol AP242 – a replacement for AP214 and AP203.

In AP242, traditional catalogues can be represented by product classes as the backbone of the digital catalogues. Each product class is a customer-oriented set of similar products that can be characterized with specifications (ISO/TS 10303-1103). Properties with values or value ranges can be associated with the product classes for further characterization. Based on an external classification framework for PLib (ISO/TS 10303-1291), the semantic ontology standardized in, e.g. ISO 13399, can be reused to classify the product classes with properties in AP242. Syntactic unification relies on data sets in p21 (ISO 10303-21) or p28 XML (ISO 10303-28) governed by a modeling language EXPRESS (ISO 10303-11). Thus, this solution unifies data structure, semantics, and syntax for cutting tool catalogue information management.

A conceptual description of the modeling approach is illustrated in Figure 2. The product is a basic element in these APs. Products of a certain product class are characterized by product specifications which indicate configurations of the

products with combinations of specifications. The specifications are categorized and associated with product classes. Semantic information is referenced from classes and properties externally defined by PLib-based product dictionaries. Properties with values or value ranges are associated to the product classes, so that both types of items can be classified respectively by the PLib classes and the PLib properties. Although association between the PLib classes and the PLib properties can be looked up in the dictionaries, this relationship is recommended to be included in this model to complete the semantics.

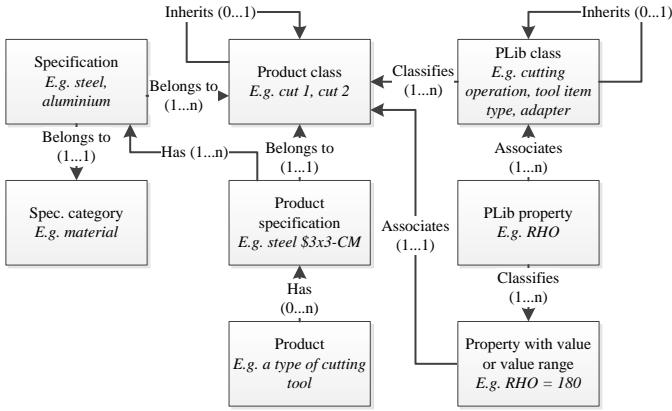


Figure 2. Conceptual model of product catalogue

Figure 3 illustrates a STEP AP242 data excerpt for representation of the backbone of a vendor catalogue. As mentioned before, the product class is an identification of a customer-oriented set of similar products. The vendors can set unique IDs, names, hierarchical levels, versions, and descriptions for the product classes. Product concept relationships specify relation types and relation sequences (ISO/TS 10303-1103).

Modeling of the classification is mainly based on a previous research for cutting tool information modeling [14] where STEP AP242 was proposed as an alternative way to replace ISO 13399-1 for cutting tool information modeling. The dictionary based on PLib was reused to classify products, features, and properties (ISO/TS 10303-1291) in the previous research. The same principles are employed here for the classification of product classes and properties with values or value ranges. Figure 4 illustrates this approach, taking classification of a product class of converters as an example. Properties such as a rotation angle of 180 degree can be associated with the product classes. The properties can be represented more dedicatedly if necessary, for e.g. dimensions and linear distances. The PLib-based cutting tool dictionary in ISO 13399 is referenced in this data model to semantically describe the product class and the property.

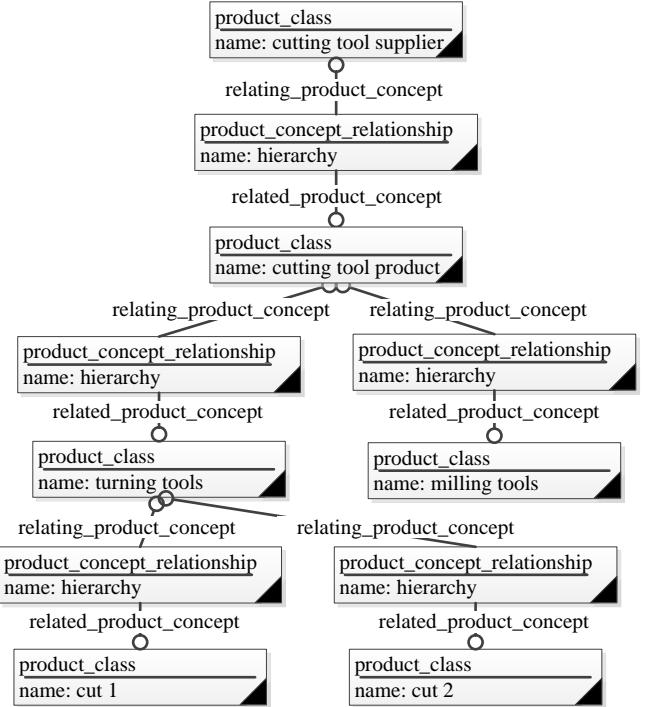


Figure 3. Example of catalogue modeling in AP242

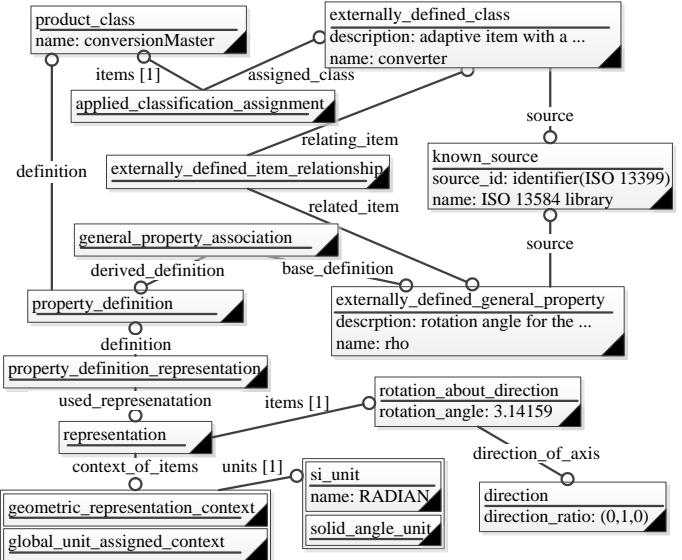


Figure 4. Example of catalogue classification modeling in AP242

In detailed levels of the catalogues, product classes should be characterized by specifications, so that variants of a product class can be discriminated from each other. In Figure 5, product concept features representing the specifications are grouped by product concept feature categories to complete semantics of the specifications. More comprehensive description of this modeling principle can be found in ISO/TS 10303-1103.

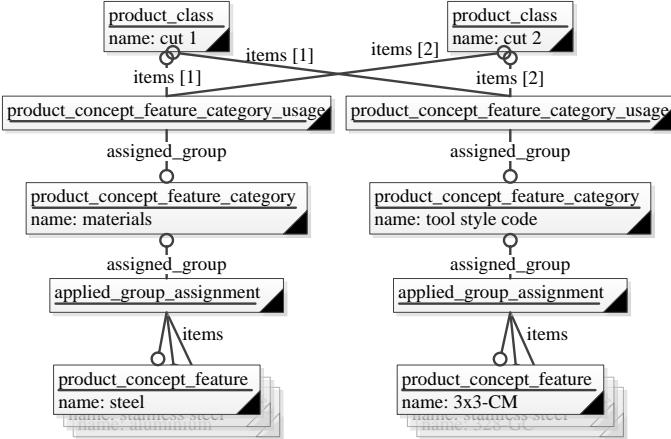


Figure 5. Example of specification modeling in AP242

This research designs not only modeling of catalogues, but also modeling of the readers' queries, in a form of customer view representation (Fig. 6). Each characterized class represents a node in the hierarchy, which can be characterized by properties with values and value ranges (ISO/TS 10303-1111). A root node is always necessary to construct a hierarchy. In the case of Figure 6, it is a node for "all products". Then the presented model shows that readers are only interested in two sets of cutting tools: face or end milling tools and tools not for milling.

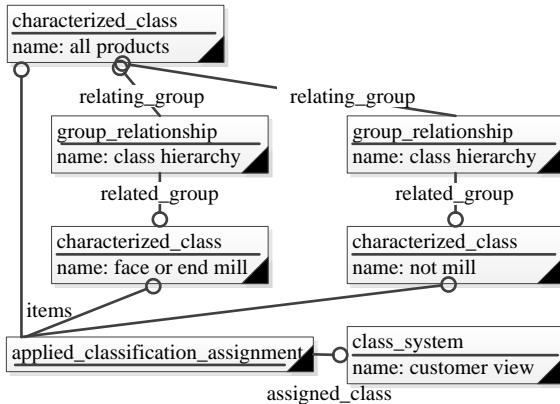


Figure 6. Example of customer view modeling

However, naming of classes should not be taken as criteria to search desired products from a catalogue. In a real application of the catalogue reader, catalogue data sets from the cutting tool vendors should be processed and filtered according to the association based on PLib and set theory, rather than the name attributes of the characterized classes. The schema of set theory (ISO 10303-54) is not included in the current version of STEP AP242, but this integration can be easily done within the modularized structure of this new application protocol. Hence this part of the data model is proposed as an extension of AP242.

Four types of operators from the set theory schema are reused in this research: same membership, complement,

intersection, and union. These operators are considered enough to facilitate the queries from the readers. The hierarchy of the customer view can be specified by the class system defined previously (Fig. 6) of which each node is a resultant of a set theory expression. In most cases, operands of such expressions are semantically standardized in the PLib dictionary. Customers can specify more complicated expressions involved multiple operators, e.g. union (turning, intersection (drilling, milling)). In such case, supplement classes should be included to complement the expressions. Figure 7 illustrates the representation of the set theory expressions for the previous example.

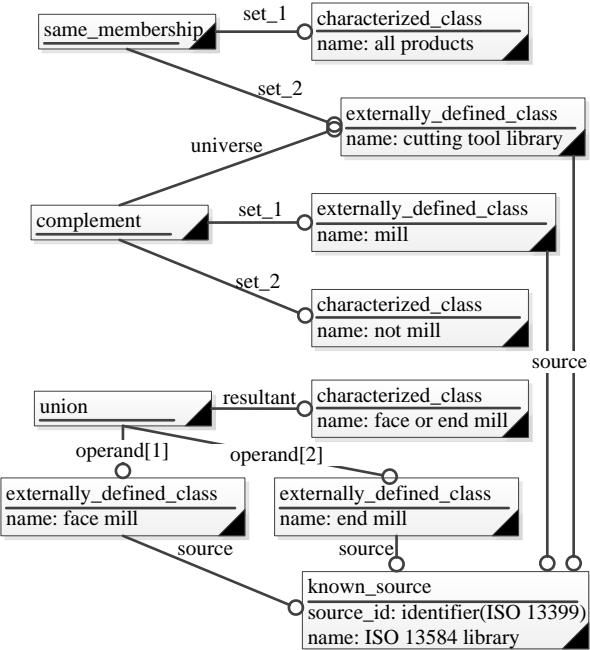


Figure 7. Example of set theory modeling

The modeling of the customer view with set theory expressions facilitates the exchange of presentation of catalogue data. The presentation can be configured by particular individuals who read the catalogues, such as buyers, users of CAM systems, and operators of CNC controllers. It can also be customized by particular organizations for specific purposes e.g. an organizational standard that is shared by dedicated readers. Of course, the readers can skip this customization of the presentation if they want to view the catalogues structured by vendors, which is similar to the traditional way but standardized.

MAPPING PRINCIPLES

Three types of software applications are designed to implement this solution (see Figure 1). Semantic mapping between vendors' products, readers' requirements, and the standard dictionary should be performed by the applications. Guidance for procedures, constraints, and recommendations of different types of mapping should be clarified.

Creation of the vendor catalogues

The vendor catalogue is a tree structure of which each node is a product class defined by vendors. A product class is a set of products which can be classified and optionally specified by classified properties with values or value ranges. The vendors define a catalogue in a software system, the vendor catalogue creator, and store it as any types of STEP data sets in AP242. The data sets are a central element to enable the product information communication. Therefore, vendors are most responsible for successful information exchange. Constraints on how to create the catalogue models should be identified and integrated into the software system. This research adopts class inclusion relationship (as used in ISO 13584-42) as the fundamental inheritance nature of this tree structure: a parent is a union of its children, i.e. a parent product class should and can only possess all products of its children product classes. This definition leads to other constraints of the vendor catalogues:

- A product class can only have one or zero parent, i.e. single inheritance;
- Products can be possessed by leaf product classes only, and non-leaf product classes obtain the possession of the products from children automatically;
- If a product class is classified by one or more classes, its parent classes cannot be classified by the sub classes of these classes;
- If a product class has one or more properties with values or value ranges, the properties (if existing) of its parent classes should have same or larger ranges and such properties should be identically classified if necessary.

Unlike the constraints, recommendations are not mandatory to be followed by the authors, but they are important to reduce workloads and human errors. The recommendations are proposed based on previous constraints:

- Only leaf product classes are classified to avoid conflicts between parents and children;
- The classification with ISO 13399 is as specific as possible, e.g. using “end mill” rather than “mill” if accurate for all the current and future possessing products.

Following the constraints and recommendations to create the catalogue models, a recommended procedure to create data models of the vendor catalogues is proposed:

1. Create the product classes for a set of similar products and set names, IDs, versions, hierarchical levels, and so on;
 - The names are free to set, but the IDs should be unique.
2. Establish hierarchical relationships between product classes;
3. Set necessary specifications to the leaf product classes and categorize these specifications;
4. Set products to the leaf product classes and associate them to the specifications if there is any;

- E.g. A product is a type of product class with a combination of two specifications (Block Tool (BTS01) and 80 °rhombic (C))

5. Classify the leaf product classes with the dictionary as specifically as possible;
6. Classify the non-leaf product classes if really necessary;
7. Export the catalogue model as a STEP AP242 data set.

Creation of the customer view

The hierarchy of the customer view represents requirements of specific readers for customized presentation of the vendor catalogues. The customer view is supposed to be customized with the customer view creator and be utilized in the catalogue reader. The nodes are filters of the product classes of the vendor catalogues. Each node of this hierarchy is a representation of a query for a desired product set, which is associated with a set theory expression of PLib classes and, optionally, relevant properties with values or value ranges. To present one or more vendor catalogues in a customized way, each node of the customer view in the catalogue reader refers to zero or more product classes according to the result of the corresponding set theory expression. The models of the customer view are transferred from the customer view creator to the catalogue reader as AP242 data sets. The models of the customer view can be archived and exchanged by end-users of the catalogue reader.

Constraints of the creation of the customer view are fewer than vendor catalogues so that readers have more freedom to present the catalogues in a desired way. Set theory expressions of all nodes of the hierarchy are free to set regardless of the configuration of parents and children. The four operators of set theory from ISO/TS 10303-1210 are tailored for readers: equal, intersection, union, and complement. As a convention for standardized computer interpretable models, these concepts are preferably hidden from end users of software. A user interface should be designed to help users understand and use the operators and the expressions based on set theory.

Since the tree structure natively exists in the customer view, the relationship between super and sub nodes should be coordinated by the catalogue reader. Authors of the customer view should provide suggestions of how to configure the catalogue reader to present a specific view. The configuration of the presentation in the catalogue reader is introduced in the following section.

Presentation principles of the catalogues

The catalogue reader is the core application that should be used most frequently among the three. As a replacement of the traditional catalogues, standardized catalogues are presented to readers with this application according to the customer view. Semantic information of the catalogues is mapped to the user interface of the catalogue reader filtered by the customer view. End-users of this application are not limited to readers of traditional catalogues, i.e. purchasing specialists of product user companies. With little knowledge on the catalogue structure, production designers, process planners, CNC operators, etc.

will benefit from such application. Required information can be specified and acquired by certain types of customer view.

The basic function of the catalogue reader is to filter the product classes to the nodes of the customer view based on set theory. The filtering logic should also be tuned in the catalogue reader to control the influence of parents and children for each node of the customer view. The basic procedure of an execution of the catalogue reader is:

1. Ask users to select catalogues and a customer view;
2. Ask users to select a filter principle: simple, upwards populating, or downwards populating (see explanations in the step 3.5);
3. Filter product classes for a certain node in the customer view;
 - 3.1. Traverse all product classes in the catalogues;
 - 3.2. Get classifications and properties ranges of all the product classes;
 - 3.3. Match the classification and properties ranges with the criteria of the node of the customer view, its child nodes, and its parent nodes;
 - 3.4. Three candidate sets of the product classes for this node can be generated:
 - Set A: The set filtered by the criterion of this node
 - Set B: The set filtered by the intersection of the criteria of all its parent nodes
 - Set C: The set filtered by the union of the criteria of all its child nodes
- 3.5. According to the user configuration of the catalogue reader, three possible presentations for this node are:
 - The simplest result: A
 - The result of upwards populating: $A \cap B$
 - The result of downwards populating: $A \cup C$
4. Repeat step 3 for all the nodes and generate a comprehensive presentation for users.

To sum up, two applications help the vendors and the customers to perform the mapping from realities to the local hierarchies according to the global ontology, a semantic dictionary. This unification makes a core application (the catalogue reader) able to automatically present the two local hierarchies in a unified way.

A basic principle of this solution is that the vendors are much more responsible to maintain reliable information exchange, and the customers have more freedom to customize the presentations. Therefore the creation of the catalogue models should be performed with more constraints than the customer view models. Although such constraints are reasonable results, or even common sense as for the data structure of a hierarchical tree, authors of the catalogues may forget these constraints. Therefore a software system to guide the creation is important to avoid failure of data exchange. In the next section, prototype implementations are introduced briefly to verify the principles for information modeling and exchange.

VERIFICATION WITH IMPLEMENTATIONS

With the help of a local cutting tool vendor, a demonstrator is achieved with prototype implementations for the three applications introduced in this solution (see Figure 1). The vendor provides a part of its catalogue in an Excel spreadsheet. The data is structured by the vendor and refers to the ISO 13399 cutting tool dictionary. Therefore at first a prototype is developed to translate the Excel data set to an AP242 data set, acting as a vendor catalogue creator. To further test the developed prototype, a catalogue from a website of another local vendor is manually documented in another Excel data set with the same structure and translated as another AP242 data set. Therefore, two different catalogues can be compared and combined in the catalogue reader. On the other hand, a simplified prototype of the customer view creator is created to help users customize and archive the presentation of the catalogues. A prototype of the catalogue reader, as the core implementation, is developed to present the vendor catalogues in the customer view. Hence, the whole prototype system can be illustrated as Figure 8.

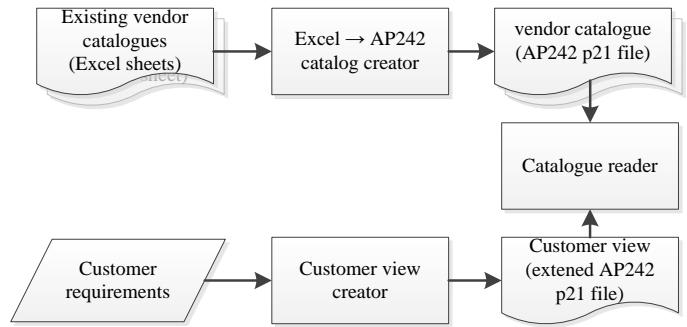


Figure 8. Implementation of prototypes

As the most important implementation, the catalogue reader is developed to track, parse, and filter data according to three types of hierarchies: the vendor catalogues, the customer view, and the standardized dictionary. These hierarchies are modeled in STEP p21 data sets based on different schemas. Effective coordination of various types of data sets and schemas is required for this implementation.

Involving coordination of two schemas and two or more data sets, the catalogue reader is the most complicated application in this demonstrator. Figure 9 illustrates the system design of the catalogue reader. As stated before, a major challenge of this prototype is to integrate the PLib information with AP242 by coordinating two types of standard data sets. The two types are governed by two schemas, and therefore processed by two SDAI (Standard Data Access Interface) libraries for programming. Therefore, the two programs (the PLib parser and the catalogue presenter) are designed relatively independent so that simultaneously management of multiple sessions of SDAI can be avoided. The PLib parser retrieves relevant information from the dictionary and keeps it in temporary memory. Then the catalogue presenter integrates the

dictionary information as semantic references to filter the product classes based on the customer view.

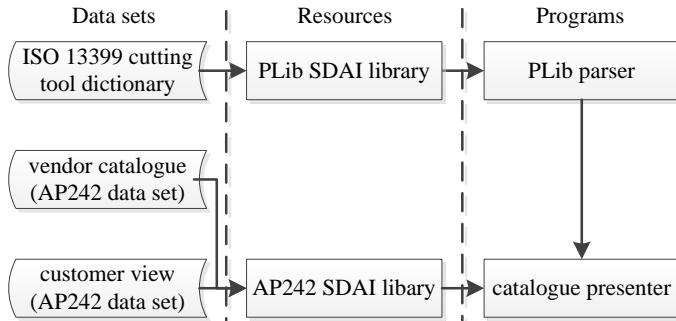


Figure 9. System design of catalogue reader

It is essential for the catalogue reader to effectively present the related product classes, products, and product specifications for each node of the vendor catalogues or the customer view. Hence, the interface of the catalogue reader is designed for the first version of the prototype as in Figure 10. In the left part of the interface, the vendor catalogues and the customer view are presented in tree structure. Users can select nodes from vendor catalogues to view associated specifications and products. Also, they can select nodes from the customer view to filter the required product classes according to the standardized dictionary, and get a product list according to certain types of

customer requirements more efficiently. Note that this prototype with the graphical interface only demonstrates most necessary elements for the catalogue readers. Real applications should be integrated with ERP, PLM, and even CAM systems of the cutting tool users.

CONCLUSIONS

ISO 13399 provides a solution to represent cutting tools with classifications based on an ISO 13584 dictionary. However, representation of product families in terms of product catalogue is not supported in ISO 13399. Industry requires an efficient solution for cutting tool catalogue communication extendable for tooling services in process planning. This research presents a standardized solution mainly based on STEP AP242 to meet the requirements. Principles on information modeling and mapping are identified to describe the solution. Prototypes are developed to verify the solution together with industrial partners. However, collaborative effort in standardization community is needed to achieve further results and complete a strong solution. Since AP242 acts as the data model for general product information, this solution is promising to fulfill similar requirements for other types of products.

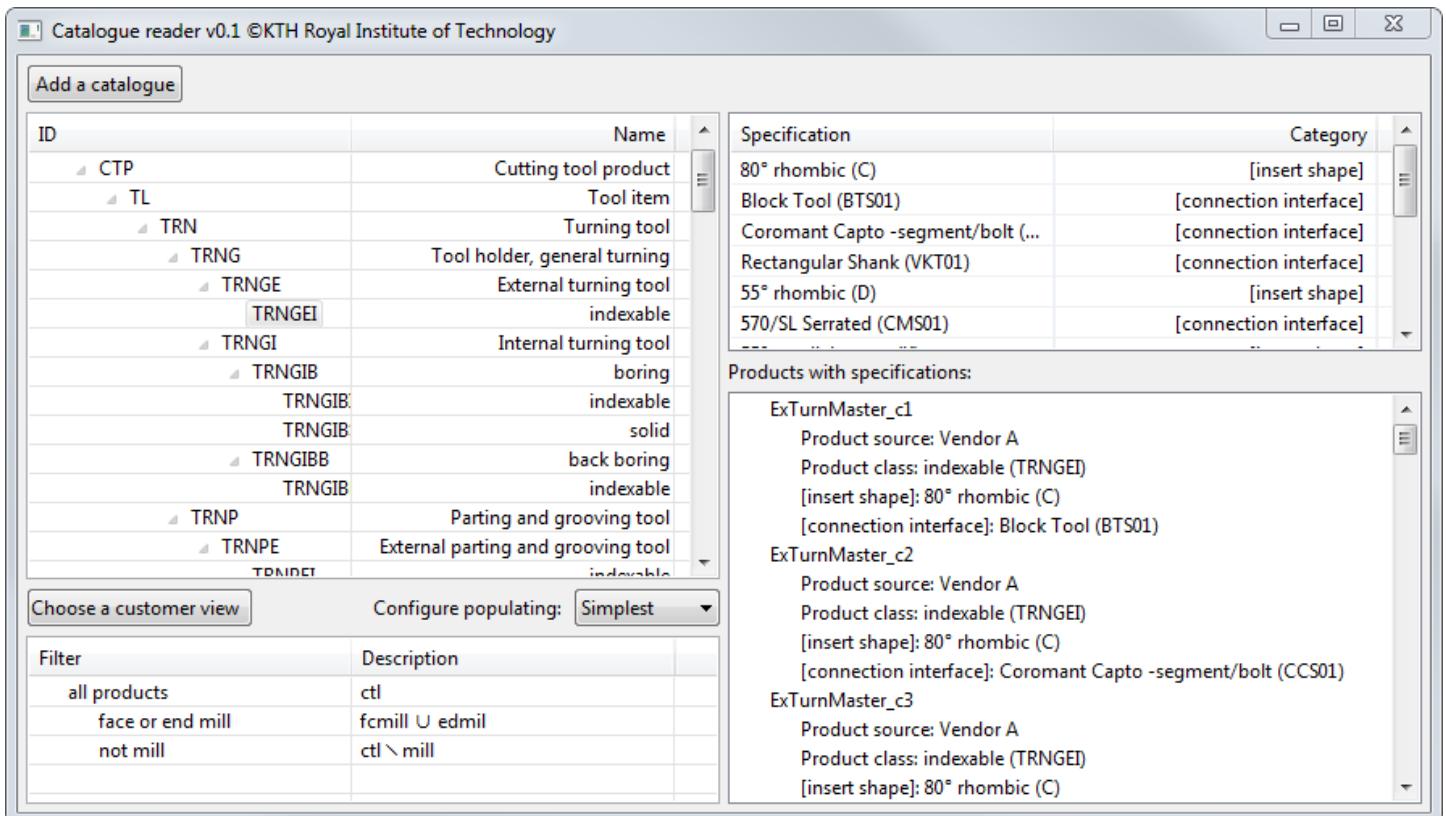


Figure 10. Interface design of the prototype catalogue reader

ACKNOWLEDGMENTS

We are grateful for the support from VINNOVA and XPRES (Initiative for excellence in production research), and for fruitful discussions with members of ISO TC184 SC4 WG3 T24 and industrial practitioners from Sandvik Coromant.

REFERENCES

- [1] Lundgren, M., 2013, "FFI Feature Based Operation Planning Technical Report," KTH Royal Institute of Technology, Stockholm.
- [2] Xu, X., 2012, "From Cloud Computing to Cloud Manufacturing," *Robotics and Computer-integrated Manufacturing*, 28(1), pp. 75-86.
- [3] Pierra, G., Sardet, E., Potier, J. C., Battier, G., Derouet, J. C., Willmann, N., and Mahir, A., 1998, "Exchange of Component Data: the PLIB (ISO 13584) Model, Standard and Tools," *Proceedings of the CALS EUROPE*, 98, pp. 160-176.
- [4] Nyqvist, O., 2008, "Information Management for Cutting Tools," Ph.D. thesis, TRITA-IIP-08-03, KTH Royal Institute of Technology, Stockholm.
- [5] Sivard, G., 2001, "A Generic Information Platform for Product Families," Ph.D. thesis, TRITA-IIP R-00-5, KTH Royal Institute of Technology, Stockholm.
- [6] Pierra, G., 1997, "Intelligent Electronic Component Catalogues for Engineering and Manufacturing," *Proceedings of the International Symposium on Global Engineering Networking*, pp. 331-352.
- [7] Albrecht, C., Dean, D., and Hansen, J., 2005, "Marketplace and technology standards for B2B e-commerce: progress, challenges, and the state of the art," *Information & Management*, 42(6), pp. 865-875.
- [8] Bellatreche, L., Dung, N. X., Pierra, G., and Hondjach, D., 2006, "Contribution of Ontology-Based Data Modeling to Automatic Integration of Electronic Catalogues within Engineering Databases," *Computers in Industry*, 57(8), pp. 711-724.
- [9] Nyambayo, J., Faraj, I., and Amor, R., 2000, "Product Libraries—Technology Review," *Proceedings of the 3rd European conference on product and process modelling in the building industry EC-PPM'2000*, pp. 267-274.
- [10] Amor, R., Jain, S., and Augenbroe, G., 2004, "Online Product Libraries: the State-of-the-art," *Proceedings of the CIB Triennial*, Toronto.
- [11] Sandvik Coromant, 2013, "GTC Guidelines," <http://www.sandvik.coromant.com/SiteCollectionDocuments/pdf/en-gb/generic-tool-classification.pdf>
- [12] Siemens, "EMO 2013 – Siemens PLM Software," 2013, <http://www.siemens.com/press/pool/de/events/2013/industry/dri-dr-technologies/2013-07-emo/backgrounder-emo-plm-e.pdf>
- [13] Wu, D., Greer, M., Rosen, D., and Schaefer, D. 2013, "Cloud Manufacturing: Strategic Vision and State-of-the-art," *Journal of Manufacturing Systems*.
- [14] Li, Y., Hedlind, M., Kjellberg, T., and Sivard, G., 2013, "Cutting Tool Data Representation and Implementation Based on STEP AP242," *Smart Product Engineering*, Springer, Berlin-Heidelberg, pp. 483-492

Paper D

Li, Y., Chen, D., Kjellberg, T. J. and Sivard, G. (2014b). User friendly development architecture for standardised modelling: STEP toolbox, *International Journal of Manufacturing Research*, Vol. 9, No. 4, pp. 429-447. DOI: 10.1504/IJMR.2014.066663.

User friendly development architecture for standardised modelling: STEP toolbox

Yujiang Li*, Danfang Chen,
Torsten J.A. Kjellberg and Gunilla Sivard

Department of Production Engineering,
KTH Royal Institute of Technology,
Brinellvägen 68, Stockholm, Sweden
Email: yujiang@kth.se
*Corresponding author

Abstract: Standardised information modelling is currently an important solution of enhancing the interoperability of computer-aided technology (CAx) systems in the manufacturing industry. The success of this solution is critically dependent on the complexity of data structure and implementation methodology. This study presents development architecture, STEP toolbox, which enables users to implement information standards via a simplified process with minimised knowledge requirements. This study starts with an analysis of relevant user groups for different types of tasks and knowledge requirements. Then STEP toolbox is presented as the simplified development architecture that consists of conceptual modelling and an object-oriented application programming interface (API). It is more convenient for typical developers than the conventional implementation methodology. Based on results from test cases for implementing three prototypes, a two-month learning process required for average developers of the information standards is expected to be eliminated.

[Received 22 April 2013; Revised 5 December 2013; Accepted 11 February 2014]

Keywords: information standards; interoperability; application programming interface; API; computer-aided technology; CAx; user groups.

Reference to this paper should be made as follows: Li, Y., Chen, D., Kjellberg, T.J.A. and Sivard, G. (2014) 'User friendly development architecture for standardised modelling: STEP toolbox', *Int. J. Manufacturing Research*, Vol. 9, No. 4, pp.429–447.

Biographical notes: Yujiang Li has been working as a Doctor student in Computer Systems for Design and Manufacturing, at KTH Royal Institute of Technology since 2011. He received his MSc in Production Engineering and Management at KTH in 2011, and BSc in Industrial Engineering at Tsinghua University, Beijing, in 2009. Currently, he focuses on the methodology and implementation in model-driven CAx system integration, based on ISO standards.

Danfang Chen received her PhD in Production Engineering at KTH Royal Institute of Technology since 2012. Afterwards she joined XPRESS – Initiative for Excellence in Production, a Swedish national research programme in production engineering as a post doc. Her main research focuses are on information management and sustainability issues for factory planning and design.

Torsten J.A. Kjellberg received his Doctor of Science in Technology, KTH. He is the Head of IVF Department of IT for production in Stockholm in 1977. He is an Appointed Docent/Associate Professor in 1986. He is the Head of the subject in 1994 and Professor 1996 in Computer Systems for Design and Manufacturing. He is the Head of Research Unit on Computer Systems in Production, at the Swedish Institute for Production Engineering Research, IVF and KTH 1977–1990. He is a Scientific Adviser to IVF 1990–2001. He created and led a big collaborative research project with industry on geometric product modelling, GPM 1979–1983. He leads the research programme on CADCAM in Sweden during the 1980s – two five-year programmes. His heading research in developing standards for small tools, ISO 13399, in cooperation with and financed by Sandvik Coromant and Kenna Metal USA, and the ISO working group 34.

Gunilla Sivard has been with the Department of Industrial Technology and Management, at KTH Royal Institute of Technology since 2004 and as a Group Leader of the research group Computer Systems for Design and Manufacturing since 2010. Her research interest is in product modelling and modelling of product development information. She received her MSc degree in Electrical Engineering and PhD in Mechanical Engineering from KTH, Stockholm Sweden. She has investigated in the areas of constraint-based configuration, modularisation, information modelling, validation and evaluation issues basically in the domain of mechanical products.

1 Introduction

Industry currently uses various types of software for digital manufacturing and digital factory tools to design and verify concepts, models, and details related to products, production, and manufacturing resources. It is strategic and necessary for an enterprise to use specialised software for different business functions (Lui et al., 2011). This phenomenon of the high diversity of devices and applications is also significant for collaborations within supply chains, which results in a large number of interoperability issues (Tolio et al., 2010). Reliable information exchange and sharing have been frequently proposed as a means of bridging the gap between specialised data models for disparate digital tools.

A system-neutral data format is critical for realising interoperability for CAx systems (Mokhtar and Houshmand, 2010). This approach requires formal information models to represent facts, concepts, or instructions with domain-specified requirements (ISO 10303-1). It is usually presented as information standards for industrial data. This study focuses on the information standards specified as computer-interpretable representations of general industrial product information for data exchange formatted in EXPRESS (ISO 10303-11; Schenck and Wilson, 1994). Such standards, including ISO 10303 standard for the exchange of product data (STEP) and related standards (e.g., ISO 13399, ISO 13584, ISO 15531, and ISO 16739), have become a family of system-neutral data exchanges. The family of standards can support a complete and seamless cycle of product data exchange from requirement analysis to delivery (Xu et al., 2005). A portion of STEP has been used as a major data exchange format for computer-aided technology (CAx) systems. In 3D geometrical modelling, two application

protocols of STEP, AP203 (ISO 10303-203) and AP214 (ISO 10303-214), are commonly implemented in most major CAx systems (Xu et al., 2011). This family of standards has potential applications in many areas (Lau and Jiang, 1998; Pratt, 2005; Campos and Xu, 2010; Khaled et al., 2010; Valilai and Houshmand, 2010).

However, only a small fraction of the vast scope of this family of standards has been implemented in industry. Observations (see Sections 2 and 3) show the lack of practical realisation of the standards primarily because of the relative complexity of its implementation methodology. Loffredo (1999) summarised that a conventional development process, based on the Standard Data Access Interface (SDAI, ISO 10303-22), includes:

- 1 the selection of APs
- 2 the selection of development tools
- 3 the selection of a data model binding methodology
- 4 system design
- 5 development
- 6 model validation
- 7 deployment.

In this process, developers need substantial knowledge for information modelling, software engineering, specific industrial domains, and, optionally, CAx systems. In turn, high demands are placed on individual workloads and development cycle time.

The objective of this research is to simplify the conventional development process and to reduce the knowledge requirements of developers by adopting the programmer-friendly development architecture known as STEP toolbox. This architecture provides a platform with which developers can easily access and manipulate datasets of the information standards. An object-oriented application programming interface (API), STEP toolbox API, is created in a way that programmers are familiar with. The complicated SDAI operations and EXPRESS-based data structure are encapsulated within the API. Only concepts that are required for specific information domains are shown to developers. Unified modelling language (UML) class diagrams can be used to present these concepts to the end-users of this API. Conceptual models in a form of ontology are used to control simplification of information models and construction of the API. Therefore, the conceptual models are a mapping bridge between the information standards and the API. Developers from academia and industry can use this solution to focus on their own priorities, instead of spending significant amounts of time learning standards-related knowledge. Another benefit is elimination of some stages of the SDAI-based development process, such as the selection of a binding methodology and model validation.

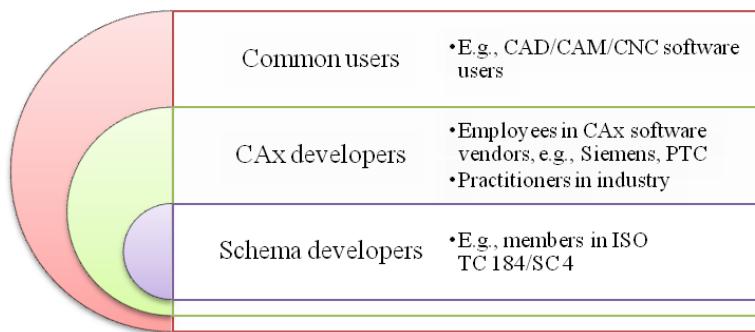
In Section 2, different levels of user groups of information standards are identified to target users potentially benefiting from this research. Then, related projects and products for these users are reviewed in Section 3. Structure of the toolbox and its underlying principles are described in Section 4. A data model mapping strategy is developed to make data models of the information standards friendly. A prototype API is developed which is based on the data model mapping with programmer-friendly documentation. In

Section 5, this mapping strategy is exemplified using kinematics. In Section 6, this new architecture is validated using three test cases performed in academia and industry.

2 User groups of information standards

STEP toolbox simplifies the use of the information standards for practical implementation. Potential users must be investigated to identify requirements on the toolbox. Three levels of users of the information standards are identified in this study (Figure 1). These user groups have different types of tasks and different levels of knowledge of the information standards. The first level consists of typical CAx users who use standard datasets, such as STEP p21 (ISO 10303-21) or p28 (ISO 10303-28) files, for data exchange between systems. These users may only know major purposes and functions of the information standards. The second level is a subset of the first level, and they have additional knowledge requirements of the information standards. Users at the second level are developers who implement CAx-related applications based on the standards. Employees of CAx software vendors may be required to programme standard-based software to facilitate communication between data formats. Practitioners in industry or academia may implement the information standards to support CAx tasks in internal and external collaborations. The third level consists of schema developers who are familiar with core structure of the information standards and can contribute to schema development based on EXPRESS language. Schema developers are usually most knowledgeable (of all the users) about the standards or a particular portion of the standards. They should also know how to implement the information standards and thus form a subset of the CAx developers.

Figure 1 Users of information standards (see online version for colours)



The main purpose of the information standards is to solve the interoperability issues of typical users (i.e., at the first level) who use all types of CAx systems. A significant amount of work has been performed since creation of these standards to improve their accessibility to typical users. Various application protocols and implementation strategies are available for different data exchange requirements. On the other hand, related researchers can easily observe complicated tasks of schema developers (i.e., at the third level), which has motivated many studies (e.g., Danner et al., 1991; Burkett and Yang,

1995; Goh et al., 1996; Feeney, 2002; Jardim-Gonçalves et al., 2005; Maier and Stumptner, 2007). However, the CAx developers (i.e., at the second level), who struggle to implement the standards, have often been neglected.

This study focuses on the CAx developers' practices. These developers may be professional programmers who develop standards-related software for themselves or be end-user programmers (Ko et al., 2011) who develop applications to support their professions or recreations. These developers are the only frequent readers of standard documents in practice and have to dedicate considerable efforts to learning relevant knowledge for implementation. Thus, a significant portion of the CAx developers' workload for a small application, such as a kinematic translation based on STEP AP242, will be apportioned toward understanding the standard documents and obtaining necessary knowledge for manufacturing engineering, software engineering, and CAx systems. The knowledge requirements of the standards for CAx developers even exceed the requirements for some schema developers (i.e., at the third level). A list of required documents (not including corrigenda) for such a case is given below:

- ISO 10303-1 for an overview of the documentation structure
- ISO 10303-11 for EXPRESS language
- ISO 10303-21 for STEP file description
- ISO 10303-22 for SDAI
- ISO 10303-27 for binding to Java
- ISO 10303-41 for fundamental product description
- ISO 10303-43 for representation structure
- ISO 10303-105 ed2 for kinematics
- And ISO/DIS 10303-242 for this application protocol.

In this study, it is observed that an average programmer costs over two months of full-time learning to understand and implement the information standards. The comprehensive structure of the standards enables data representations for a wide range of industrial sectors and lifecycle stages of products (Pratt, 2001) while also makes it difficult for developers to learn the standards initially. In addition to the substantial learning procedure, other practical issues in information modelling can be observed to fail the implementation, such as inconsistent instantiation (von Euler-Chelpin, 2008). Developers can interpret the information models differently because of inexperience, insufficient knowledge, or carelessness, thereby deteriorating the unambiguous usage of the standards.

The information standards for industrial data management provide enterprises with fast and reliable data exchange in internal and external communication. However, the time required for the preparation, development, and validation has become a critical obstacle to using these standards in their target areas. Both researchers and practitioners complain that the associated human resources costs make the standards unfriendly to both individuals and organisations.

3 State-of-the-art implementation of information standards

As previously stated, information standards, such as STEP, are important candidates for system neutral product data representation and exchange, notably because of unambiguous modelling of product data (SCRA, 2006). For geometric data exchange, STEP AP203 and AP214 have been implemented in almost all major CAD software, e.g., Siemens NX, CATIA, AutoCAD, and Pro/ENGINEER. However, a large portion of the standards are still far from practical implementation in many domains (e.g., shipbuilding and architecture) and from being used in implementation methods (e.g., EXPRESS-XML and EXPRESS-UML). Tassey (1999) has identified three causes of the failure of STEP development for interoperability:

- 1 unfairly sharing of economic costs and benefits
- 2 technology and global marketing risks
- 3 a lack of unbiased expertise.

Users at the second level perform development for interoperability, as previously mentioned. All of these reasons for failure are related to the considerable requirements that are placed on the individuals of this user group. An important solution to circumvent this failure is to lower the technical complexity of the standards in implementation and to increase user experiences of the CAx developers.

In this study, the implementation of the information standards refers to the development process of software applications with functions for accessing or manipulating EXPRESS-based datasets (e.g., p21 files or p28 XML files). As previously mentioned, developers typically have a heavy workload for this process, especially during the preparation stage for learning the standards comprehensively. Thus, the development architecture, STEP toolbox, is proposed in this study to considerably simplify the conventional development process.

Most implementations use existing SDAI-based development tools to save human resources and costs over manual operation of datasets and schemas based on the standards. Such tools are usually implementations of SDAI in specific programming languages that provide fundamental operations for accessing, manipulating, and validating EXPRESS-based datasets. For instance, JSDAI (provided by LKSoft) is a Java programming toolkit based on SDAI for schemas and datasets development (Klein, 2000). JSDAI includes a complete implementation of ISO 10303-27, upon which the prototype of STEP toolbox API is based. STEPcode (SC) is a similar development tool that is based on C/C++/Python and replaces a former product, STEP Class Library (SCL) (Sauder and Morris, 1995). Reeper is unique in its support of XML-based schemas and datasets that conform to ISO 10303-28. In the commercial software market, ST-developer has been developed over two decades and has been adopted in several major CAD applications for STEP translation. Other players include EDMdeveloperSeat by Jotne EPM Technology, ECCO Toolkit by PDTec, etc. Table 1 makes a simple comparison of some of the tools.

Such development tools significantly reduce the preparation work of developers but remain within the conventional development process of information standards. Developers need to understand data models, EXPRESS schemas, SDAI, and so forth very well before they can begin programming. In this study, STEP toolbox represents a

substantially departure from the conventional development process. This architecture has low knowledge and experience requirements and can be expected to significantly reduce time expenditures and workloads.

Table 1 Examples of STEP development tools

Name	JSDAI	STEPcode	ST-developer	Reeper
Creator	LKSoft	BRL-CAD, NIST, and others	STEP tools	EuroSTEP and NIST
Version	4.3.0	0.6	15.0	0.5
Availability	Free for non-commercial use	Open source	Commercial software	Open source
Language	Java	C, C++, and Python	C, C++, and Java	Ruby
Major features	EXPRESS schema compiler, SDAI library, access of p21, EXPRESS-based validation			XML-based operation on schemas and datasets
Additional features	Plugin in Eclipse, access of p28 XML file	EXPRESS formatter	Access of IGES and DXF	

4 STEP toolbox architecture design

The objective of STEP toolbox is to provide a programmer-friendly, maintainable, reusable, and extensible platform to simplify the implementation process. Programmers can use the conceptual models and the object-oriented API to build up their knowledge with a set of documents that are more similar to other types of APIs. This new generation of the implementation process is as intuitive to the user as a typical software development process, unlike with the conventional implementation methodology.

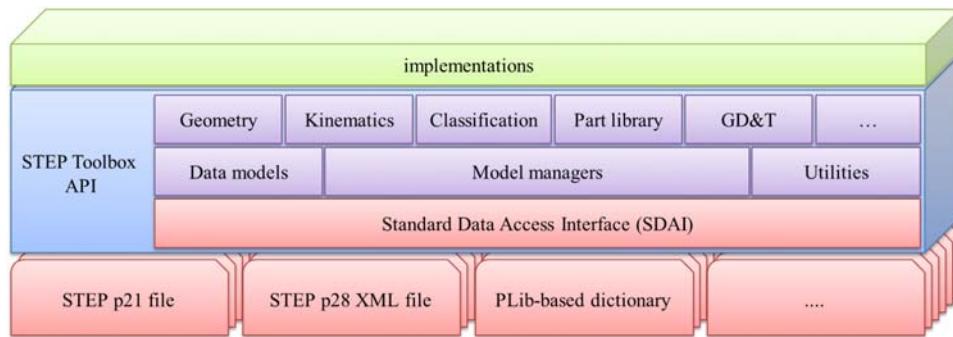
4.1 Illustration

Figure 2 illustrates the architecture of STEP toolbox in which three layers show where and how the toolbox API should be applied. The figure schematises the internal modularised structure of the API (the middle layer), the external behaviours interact with standardised datasets (the bottom layer), and prototype applications are built upon the toolbox (the top layer). The standardised datasets are governed by EXPRESS-based schemas and defined in standardised syntax. The middle layer is the core element of this architecture, i.e., STEP toolbox API. The API includes not only the modularised components developed in this study, but also a library of SDAI as a compensation for advanced usage. The top layer indicates implementations based on STEP toolbox API, which demonstrates the performance of the architecture.

In the bottom layer, the requirements for manipulating various types of standard file formats are critical for determining the external behaviours of the API. In practice, STEP p21 is the most widely used file format in the family of the standards. An alternative, p28, is introduced such that XML can be employed for both schemas and datasets. EXPRESS as a widely used modelling language enables STEP toolbox API to process

data formats for many information standards in industry, e.g., access to the ISO 13399 classification dictionary based on PLib (ISO 13584) has been previously tested using STEP toolbox API (Li et al., 2013).

Figure 2 Architecture of STEP toolbox (see online version for colours)



The API is located in the middle layer. It is the only interface that the developers need to interact with when implementing the information standards. This API provides the same functionality as SDAI for exporting, importing, and modifying datasets, albeit in a much simpler way. SDAI is kept for manipulating the datasets in the p21 or p28 file format that programmers can invoke any time. To minimise the learning process and simplify interaction between the CAx developers and the data models, the toolbox API acts as a high-level programmer-friendly API upon which CAx plugins or standalone applications can be built. A detailed description of the STEP toolbox API will be presented in Section 4.2.

In the top layer, the information standards can be implemented upon STEP toolbox API, such as plugins to tailor CAx software or standalone applications. Successful examples have been developed in different areas by authors and project partners. For instance, a kinematics translator for Siemens NX motivated this research by demonstrating the feasibility to implement the standards with commercial CAx software in other information domains than geometry (Li et al., 2011). Afterwards, STEP toolbox was initiated to support industrial project partners in CATIA-based integration development. Other examples include an optical coherence tomography (OCT)-based point cloud translator for in-process monitoring, a cutting tool classification application based on ISO 13399 and STEP AP242, an ASME/STEP translator for kinematic error data exchange, etc.

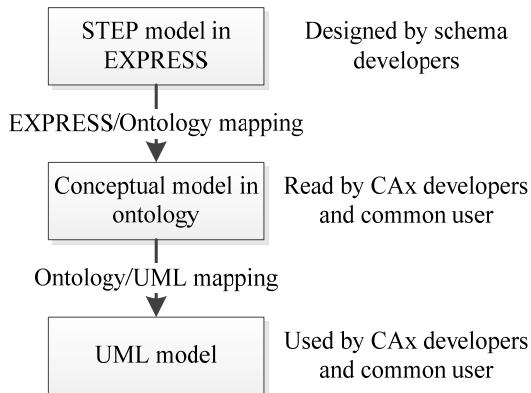
4.2 Internal design of STEP toolbox API

Figure 2 illustrates components of STEP toolbox API in the middle layer. Except for the SDAI layer, the other components represent this new generation of the STEP implementation process. The STEP model managers provide general functions to initialise, export and close a model from a p21 or p28 file. The data models represent all classes for different information concepts and their relationships, e.g., components, kinematic links, diameters, linear distances, PLib classes, and properties. The utilities are a group of tool classes that provide generally reusable functions, e.g., calculations for a coordinate system transformation.

The top level of the API consists of different modules grouping the data models and the relevant operations with interrelated classes, e.g., kinematics and GD&T. For each module, a specific manager views and modifies the data of a corresponding information type. Modularisation is important to simplify respective information models so that developers can navigate the data models and perform the necessary operations easily, because of the wide range of domains covered by the information standards. For instance, a classification application (Li et al., 2013), which is based on ISO 13399, AP242, and PLib, relies on the modules of geometric dimensioning and tolerancing (GD&T), classification, part library, and a part of geometry. For this application, developers can use the GDTManager in the GD&T module and the PLibManager in the part library module to collect available information from the datasets and the PLib-based dictionary. Then a ClassificationManager can modify classifications of the GD&T data according to user requirements.

It requires a bridge to simplify information models from EXPRESS schemas to object-oriented API. The toolbox adopts conceptual models in a simple form of ontology to describe concepts and relationships of the modularised data models. The conceptual models are generated from the EXPRESS schemas to instruct the API design. API users can easily obtain information on the data models of the toolbox from the conceptual models. Therefore, the relevant concepts of specific information domains should be easily interpretable by humans and can be mapped to the API to implement the concepts as the data models. Figure 3 illustrates this mapping, which conceals the relatively complicated EXPRESS schemas from programmers. The mapping between the EXPRESS schemas and the conceptual models should be based on a general understanding of relative domain specific knowledge. In the next section, prototypes for kinematics translation will be used to detail how to perform this mapping process.

Figure 3 EXPRESS/ontology/UML mapping



5 Data model generation

STEP toolbox includes a large amount of information in several modules even at this prototype stage. Since the modules share same design principles and similar data structure, it is not necessary to introduce all the modules repeatedly. In the remaining part

of this article, the aforementioned data model mapping process will be exemplified using kinematics.

STEP p105 (ISO 10303-105) addresses the kinematic mechanism representation. AP214 is currently the only application protocol that uses the first edition of p105. In an ongoing project, a new application protocol, AP242, is being developed, of which p105 ed2 is an important component for developing a new generation of the standardised kinematic mechanism representation (Hedlind et al., 2011). These latest standardisation results, i.e., AP242 and p105 ed2, are used in this study.

Kinematics is chosen as a representative example among the different modules because of implementation requirements of data exchange from both academia and industry (more details are provided in Section 6). As stated in Section 1, the considerable knowledge requirements and the unusual development process are significant obstacles in STEP implementation. Users also have remarkably little experience with AP242 and p105 ed2 because the related schemas have been realised quite recently. Therefore, a user-friendly solution for simplifying the implementation process should be highly valuable to developers from academia and industry.

5.1 Kinematic model based on STEP p105 ed2

STEP toolbox API adopts the new STEP components (i.e., AP242 and p105 ed2); thus, it is necessary to briefly introduce the related standardised modelling approach. Basic concepts of kinematic mechanism modelling are common to all major CAD software: links, joints, and pairs are combined to represent the mechanism. Links act as rigid or linear flexible objects in a topological structure. Each joint is composed of two links to describe topological aspects of the mechanism. For geometrical aspects, pairs describe motion constraints and geometrical associations.

Figure 4 Information modelling of a kinematic link in AP242

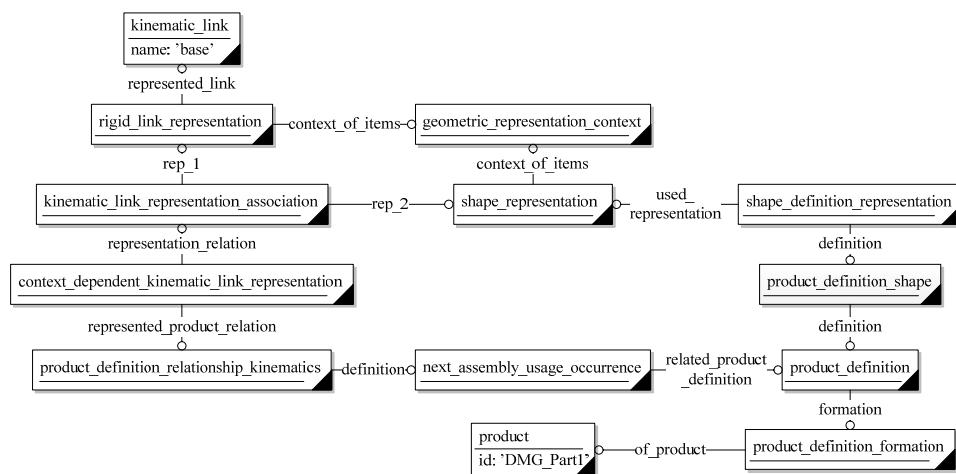


Figure 4 illustrates an example of a kinematic link that is modelled within an AP242 p21 dataset. An instance of a kinematic link is used to compose the topology of the mechanism (Figure 5). The geometrical aspect of a link can be represented by a rigid link that defines the geometric placements of each user pair. As a new feature of p105 ed2, a link is treated as a property associated with a product definition via a context dependent kinematic link representation.

Figure 5 Kinematic topological structure of a five-axis DMG70 machine in AP242

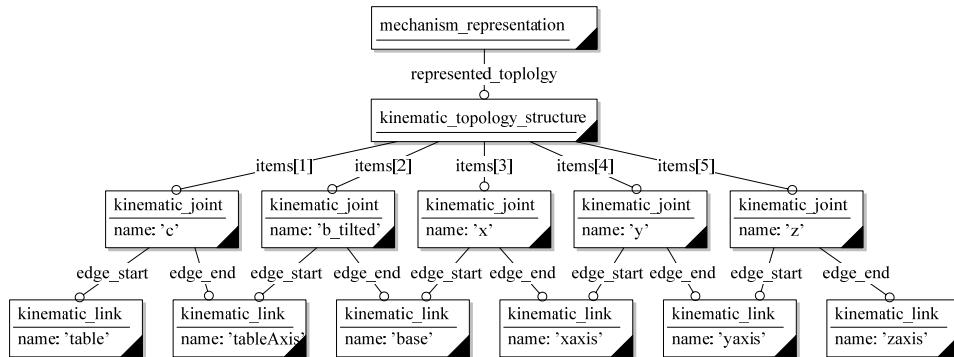
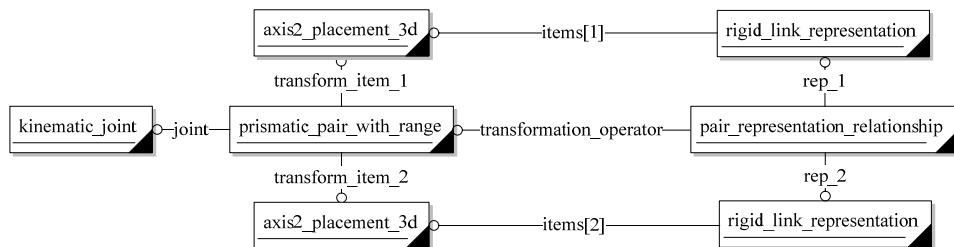


Figure 5 shows an example of the topological structure for a five-axis machine tool. Applying graph theory, a kinematic link is treated as a vertex among one or more joints which are treated as edges. In this example, the five joints represent the five axes. The instance of the kinematic topology structure is used to collect all of the relative joints and to associate these joints with a mechanism.

Figure 6 shows the geometrical aspect of a prismatic pair and its links. The prismatic pair with (a) range defines the DOF (Degrees Of Freedom) of the translation motion between two links along an axis. The range attributes of the pair are used to define the values of the lower and upper limits, if necessary.

Figure 6 Information modelling for geometric aspect of a prismatic pair



This section briefly introduces the kinematic mechanism based on STEP p105 ed2. Note that only the necessary attributes and relationships of the instantiations are discussed here. There is no detailed document of this standardised modelling methodology for developers, which easily leads to expectation that beginners may take a very long time for successful implementation.

5.2 Conceptual modelling

The conceptual models in a simple form of ontology are a bridge between the EXPRESS schemas and the API. The models regulate how concepts in the information standards can be simplified and interrelated as per implementation requirements. The models correspond to the convergence of the STEP models, the CAx concepts, and the domain-specific concepts, so that the API can be understood easily by developers.

A vocabulary that is interpretable by humans is defined in the conceptual models so that programmers have the same concepts as in the standards. Concepts in the standards are usually defined differently in various CAx systems or data formats. Therefore, a unified and general vocabulary is needed. For instance, in a kinematic mechanism of a machine, a base link must be assigned which is fixed on the ground or remains relatively static. In an AP242 dataset, such a link is referred to as a base attribute by the mechanism representation. In Siemens NX, the same concept is expressed by assigning a Boolean attribute to a link if it is fixed. In CATIA, links are not defined, but geometric parts can form a joint directly, and a base link is represented by assigning a part as fixed.

Since there are several modules, concepts should be reusable and independent of schemas. Reusability means that concepts can be shared by two or more modules. For instance, a ‘product’ in STEP is a general concept that may have one or more occurrences in a dataset. Occurrences are more useful than products in most CAx implementations; thus, the occurrence, instead of the product, is taken as a class in the ontology and is called a ‘component’. For the same product in a STEP dataset, each occurrence is set as an instance of the ‘component’ in the module of geometry and shares the same product ID as an attribute. Thus, the component can be reused easily in other modules, e.g., in the module of kinematics, a link can be associated with one component. In some information standards, one concept may be shared by different schemas in the same or different structures. As an example, the kinematic link is in different schematic structures in AP214 and AP242, but the basic concepts are defined very similarly. Therefore, users should be allowed to export a kinematic model with AP214 or AP242 based on a same data model in STEP toolbox.

Figure 7 A simplified ontology model for kinematics

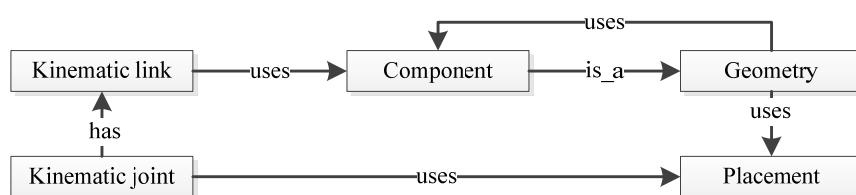


Figure 7 exemplifies a conceptual model for kinematics. Note that the properties and their facets for all classes in the model are not shown in the figure. For instance, a kinematic joint has a property that defines its motion type, such as prismatic, revolution, cylinder, etc. To make the conceptual models be easily mapped to a computer interpretable API, several simple terms are used to describe the relationships between the classes of the ontology, e.g., generalisation, aggregation, association, and dependency. The presented ontology is a combination of concepts of the kinematics and the geometry module. A component is defined as a subclass of the geometry that uses a placement to identify a

location and an orientation. Each kinematic link is associated with a component. Two links make up a joint for which the placement is used to determine both the location and the orientation. The exemplified ontology model can be extended further for other purposes, such as kinematic error modelling (Li et al., 2012).

5.3 Generation of the API

A well-designed and documented API is important for implementation. The success of an object-oriented API replies on the design of classes. In STEP toolbox API, there are two types of classes: classes of data models which represent basic elements of concepts for implementing the information standards, and classes of managers with static attributes and operations that collect information in datasets and that perform necessary functionalities. One of these managers, the STEP model manager, is a central controller for initialising, exporting, and closing the datasets. Each module has its own manager to manipulate related data.

Figure 8 A simplified UML class diagram for kinematics in STEP toolbox API

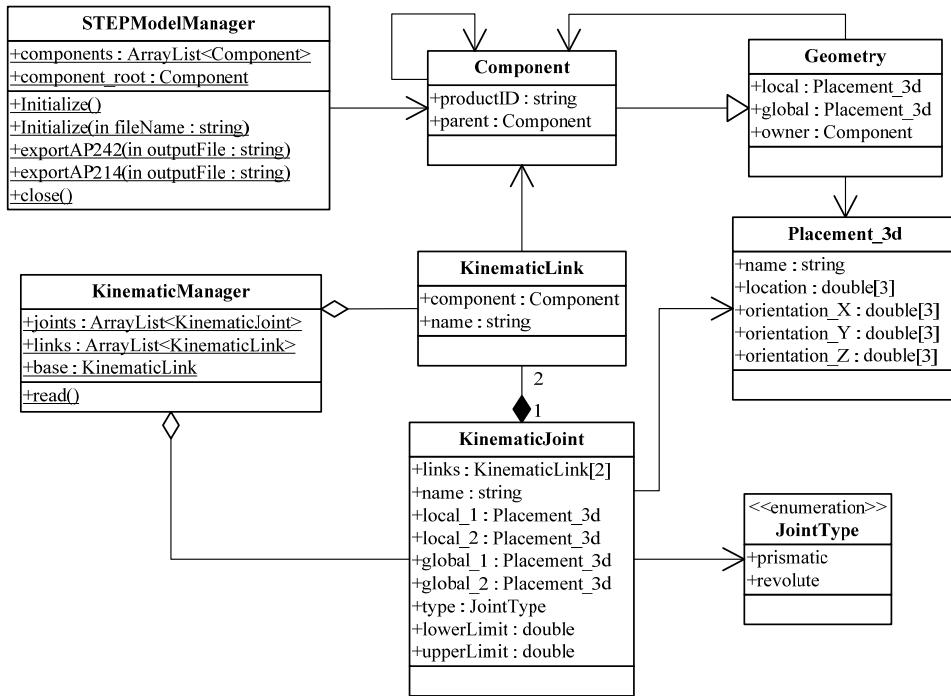


Figure 8 shows a simplified class diagram of the API for the kinematic mechanism. Note that only important classes and attributes are illustrated here. The kinematic manager collects available joints and links as static attributes so that the developers can easily manipulate these attributes. For the data model, in addition to the five types of classes mapped from the ontology (Figure 7), an enumeration set is defined for the kinematic joint type. In STEP p105 ed2, nine types of low order pairs are defined depending on different combinations of degrees of freedom. Two of the most frequently used types are

presented here as examples: prismatic and revolute. For simplicity, getters, setters and constructors for all data model classes are not shown; however, these elements should be applied correctly within the context of CAx development. For instance, in a STEP kinematic model, the placement of each link is defined in a link frame, i.e., the coordinate system of the corresponding geometric component of the link. Thus, each joint has two placements defined in two link frames. However, the joint placements may be defined within the global coordinate system in some software, e.g., Siemens NX, or within the local coordinate system, e.g., CATIA. Thus, the constructors of the joint should be defined for both local and global coordinate systems. The class diagram provides an illustration on the object-oriented API. In the following section, test cases exemplify how to use the API.

6 Test cases

STEP toolbox is proposed to facilitate programmer-friendly implementation. The API is the core element of this architecture. In this section, kinematic translations are developed and integrated with Siemens NX and CATIA. Two NX add-ons are developed in-house to compare implementations with and without STEP toolbox API. The CATIA add-on is developed externally with the API by industrial project partners. This add-on validates the expected benefits of this architecture, i.e., the elimination of the learning process and the simplification of the development process.

6.1 Kinematic translation for Siemens NX

The integration development based on Siemens NX was initialised using STEP AP214 (Li et al., 2011), from which the focus has been changed to AP242 with the evolution of the new standard. The previous application demonstrated the feasibility of CAx-integrated applications for kinematic mechanism data exchange with STEP. Since then, a project partner, a Swedish vehicle company, has required a convenient way of integrating standardised kinematic translations within its own CAx systems. Therefore, the STEP toolbox is designed and tested in-house with Siemens NX. To demonstrate this solution in a concrete manner, two AP242-based translators for Siemens NX are created and compared with and without STEP toolbox.

The system design of the first translator (Figure 9) without STEP toolbox is similar to that developed in a previous study (Li et al., 2011). This system integration is performed with NX Open API and JSDAI API. The former API is used to generate an AP214 dataset of geometry via the NX native translator and to collect data for the kinematic mechanism from an NX model. JSDAI API is used to integrate the kinematic data with the generated AP214 dataset and to export an AP242 dataset. In addition, JSDAI also provide a software package that can compile a schema to a Java class library.

Data integration, i.e., mapping for the kinematic mechanism between Siemens NX and STEP AP242, should be performed correctly. For assembly and geometry, the AP242 schema is compatible with the AP214 datasets; therefore, the AP214 datasets exported from CAx software such as Siemens NX and CATIA can be directly processed by the class library of AP242. Thus, this development does not require cross-AP schema mapping. To accomplish the data integration, developers must correctly process the schemas and the datasets using their information modelling knowledge, e.g., the mapping

from application reference model (ARM) to application integrated model (AIM). Using early binding as an example, developers must first compile related schemas of certain APs. Schematic or conceptual similarity may exist among two or more APs, developers should correctly programme against those APs. After coordinating the schemas, developers should map the concepts of the CAx systems to the information standards for accurate information translation. Correct mapping of integrations between the different information domains, e.g., geometry and kinematics, is also critical for successful data integration.

Figure 9 System design of STEP kinematic translator for NX without STEP toolbox

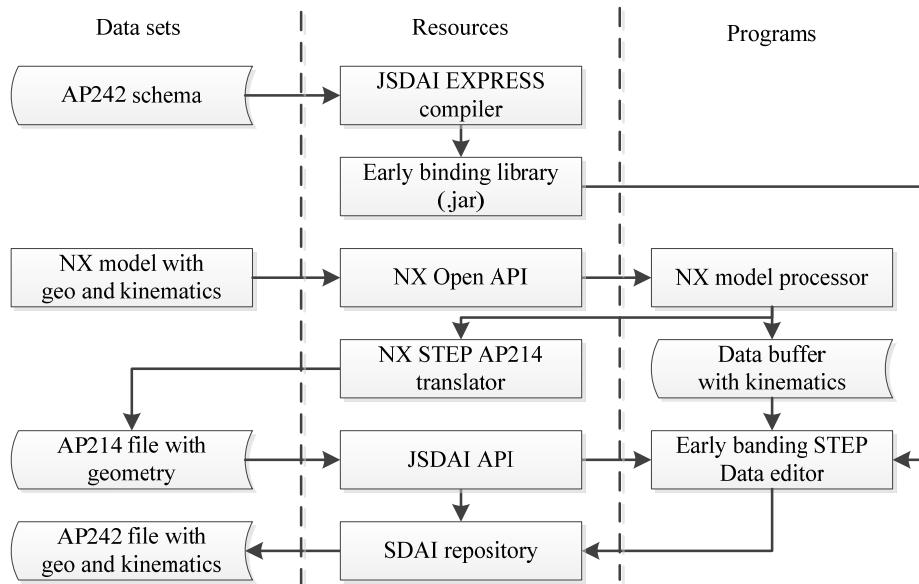


Figure 10 System design of STEP kinematic translator for NX with STEP toolbox

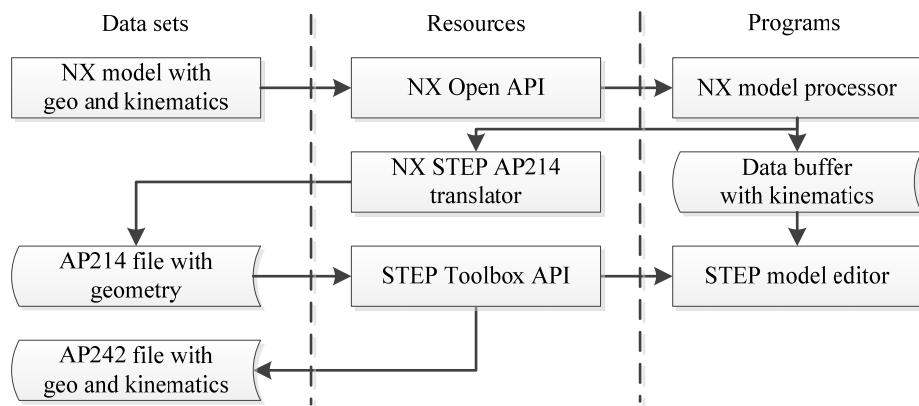


Figure 10 shows that STEP toolbox simplifies implementation which is integrated with CAx systems. Programmers can use this system, without any knowledge requirements, to

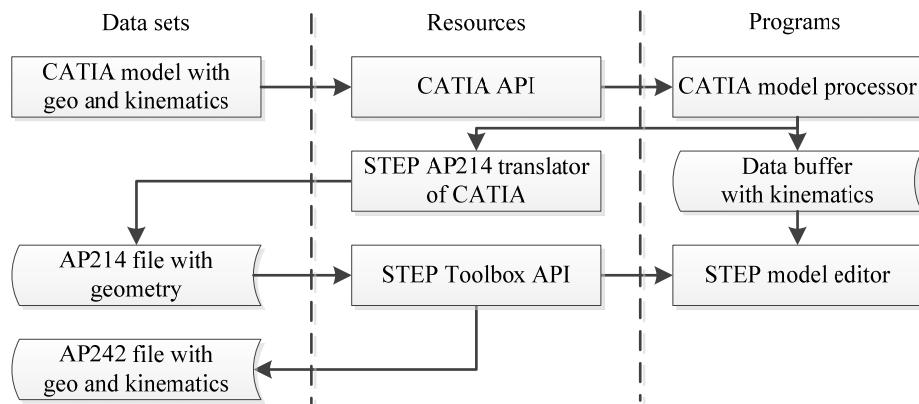
implement previous standards. Schemas are compiled into class libraries and encapsulated within the API so that users need not even know of the existence of SDAI, schemas, early or late binding, etc. Developers can control the datasets to be initialised, exported, and closed using simple interfaces that compress all of the related SDAI operations on sessions, models, repository, and transactions. All manipulations of the datasets are performed using programmer-friendly data models and managers. Although it is not a contribution of this paper, this system design is an important solution to use STEP toolbox integrating with CAx systems. This design separates the programme interacting with Siemens NX from the other programme for the standard datasets by the data buffer and can thus easily be migrated to other CAx systems with APIs in any programming language. The following section will describe how this development approach can be migrated to CATIA.

6.2 Kinematic translation for CATIA

An important objective of STEP toolbox is to help industrial developers in their own CAx environments. This section describes how the system design (Figure 10) with the toolbox API can be migrated to a different CAx system, CATIA, where it can be used with a different CAx API, CATIA VBA.

Javadoc and sample codes are two important documentation methods for sharing STEP toolbox API. For any Java-based API, Javadoc is an important tool that provides detailed instruction on each element with a structured HyperText Markup Language (HTML) framework. However, starting to programme only with Javadoc remains challenging. Therefore, sample codes are provided for more know-how knowledge on how to initialise, modify, export, and close a STEP dataset for specific information domains in practice.

Figure 11 System design of STEP kinematic translator for CATIA with STEP toolbox



An AP242-based kinematic translator for CATIA has been implemented by developers in industry with the system design shown in Figure 11. Similar to the implementation for NX, the CATIA model processor is coded with CATIA VBA API. Using STEP toolbox API, a Java program, the STEP model editor, is used to manipulate the STEP datasets. A

data buffer in comma-separated values (CSV) format is designed to bridge the VBA script and the Java program. The STEP AP214 translator of CATIA can be invoked by the VBA script to export AP214 files only with geometry. Then, STEP toolbox API can integrate the kinematic data with geometry and generate AP242 files.

Industrial developers do not learn any information from the standard documents until the end of the implementation. Similar to other typical programming tasks, Javadoc and sample codes are the only documents to look up. The test cases show that STEP toolbox enhances the implementation performance over that of the traditional SDAI approach.

7 Conclusions

The information standards have a comprehensive capability which is important for product data modelling and exchange; however, this capability also makes the documents difficult to understand by CAx developers. STEP toolbox provides a programmer-friendly solution in this context. The architecture design of STEP toolbox focuses on two aspects: structure and behaviour. The structure design encapsulates EXPRESS schemas and modularises data models. The behaviour design provides a programmer-friendly solution for interaction that practitioners can easily access and utilise. Two in-house implementations and a collaborative implementation with industry are performed to validate this solution in academia and industry. Preliminary observations of average programmers show that a two-month learning period for standard-related knowledge can be easily eliminated using STEP toolbox. The implementation process becomes simpler and more convenient for programmers.

The successful test of STEP toolbox in industry motivates standardisation of industrial product data modelling and exchange based on the information standards. Technically, the toolbox can avoid inconsistent instantiation, non-neutral definition, and information loss because the generation of dataset is encapsulated by the API and designed by experts. The programmers' tasks are simplified so that the programmers can focus on higher priority issues. For businesses, the resources spent on implementation are reduced and easier to predict because the development process becomes more similar to normal software development, and fewer experts on information modelling are needed. Therefore, industry gets motivated and prepared to perform implementation.

The STEP toolbox presented in this article is still in a prototype stage. Test cases have demonstrated its practical application, but further development is necessary to turn this prototype into a mature contribution. This paper focuses on what STEP toolbox is, who its end-users are, and how the toolbox can be used. However, practical questions remain: who should produce STEP toolbox and how should STEP toolbox be developed? Generally, it should be developed according to various practical requirements of the CAx system designers, developers, and typical users. This process requires a precise and unified understanding of lifecycle characteristics of different types of industrial products, production, and manufacturing resources. Developer-oriented usability analysis would be helpful in improving STEP toolbox further. Tasks for different communities, i.e., academia and industry in this area, should be clarified to achieve mutual benefits in the future.

References

- Burkett, W. and Yang, Y. (1995) 'The STEP integration information architecture', *Engineering with Computers*, Vol. 11, No. 3, pp.136–144.
- Campos, J.G. and Xu, X.W. (2010) 'STEP-NC-compliant machine automation to support saw blade stone-cutting machining', *International Journal of Manufacturing Research*, Vol. 5, No. 1, pp.58–73.
- Danner, W.F., Sandford, D.T. and Yang, Y. (1991) *STEP (Standard for the Exchange of Product Model Data) Resource Integration: Semantic & Syntactic Rules*, NIST National Institute of Standards and Technology, NISTIR 4528.ISO TC184 SC4WG5 N10.
- Feeney, A.B. (2002) 'The STEP modular architecture', *Journal of Computing and Information Science in Engineering*, Vol. 2, No. 2, pp.132–135.
- Goh, A., Hui, S.C. and Song, B. (1996) 'An integrated environment for product development using STEP/EXPRESS', *Computers in Industry*, Vol. 31, No. 3, pp.305–313.
- Hedlind, M., Klein, L., Li, Y. and Kjellberg, T. (2011) 'Kinematic structure representation of products and manufacturing resources', in *Proceedings of the 7th CIRP-Sponsored International Conference on Digital Enterprise Technology*, Athens, Greece, pp.340–347.
- Jardim-Gonçalves, R., Olavo, R. and Steiger-Garção, A. (2005) 'The emerging ISO10303 modular architecture: in search of an agile platform for adoption by SMEs', *International Journal of IT Standards and Standardization Research*, Vol. 3, No. 2, pp.82–95.
- Khaled, A., Ma, Y.-S. and Miller, J. (2010) 'Feature and product markup languages in service-oriented CAX collaboration', *International Journal of Manufacturing Research*, Vol. 5, No. 1, pp.87–101.
- Klein, L. (2000) *JSDAI – The Synergy of STEP and JavaTM Technology* [online] http://doc.jsdai.net/jsdai_doc/tutorial/tutorial.pdf (accessed 31 March 2013).
- Ko, A.J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., Scaffidi, C., Lawrence, J., Lieberman, H., Myers, B., Rosson, M.B., Rothermel, G., Shaw, M. and Wiedenbeck, S. (2011) 'The state of the art in end-user software engineering', *ACM Computing Surveys*, Vol. 43, No. 3, pp.21:1–21:44.
- Lau, L. and Jiang, B. (1998) 'A generic integrated system from CAD to CAPP: a neutral file-cum-GT approach', *Computer Integrated Manufacturing Systems*, Vol. 11, Nos. 1–2, pp.67–75.
- Li, Y., Hedlind, M. and Kjellberg, T. (2011) 'Implementation of kinematic mechanism data exchange based on STEP', in *Proceedings of the 7th CIRP-Sponsored International Conference on Digital Enterprise Technology*, Athens, Greece, pp.152–159.
- Li, Y., Hedlind, M. and Kjellberg, T. (2012) 'Kinematic error modeling based on STEP AP242', Paper presented at the *1st CIRP Sponsored Conference on Virtual Machining Process Technology*, Montreal, Canada, 28 May–1 June.
- Li, Y., Hedlind, M., Kjellberg, T. and Sivard, G. (2013) 'Cutting tool data representation and implementation based on STEP AP242', in Abramovici, M. and Stark, R. (Eds.): *Smart Product Engineering*, pp.483–492, Springer, Berlin/Heidelberg.
- Loffredo, D. (1999) *Fundamentals of STEP Implementation* [online] <http://www.step-tools.com/library/fundimpl.pdf> (accessed 31 March 2013).
- Lui, M., Gray, M., Chan, A. and Long, J. (2011) 'Enterprise application integration fundamentals', in Lui, M. et al. (Eds.): *Pro Spring Integration*, pp.1–14, Apress, New York.
- Maier, F. and Stumptner, M. (2007) 'Enhancements and ontological use of ISO-10303 (STEP) to support the exchange of parameterised product data models', in *Seventh International Conference on Intelligent Systems Design and Applications (ISDA 2007)*, Rio de Janeiro, Brazil, pp.433–440.
- Mokhtar, A. and Houshmand, M. (2010) 'Introducing a roadmap to implement the universal manufacturing platform using axiomatic design theory', *International Journal of Manufacturing Research*, Vol. 5, No. 2, pp.252–269.

- Pratt, M.J. (2001) 'Introduction to ISO 10303 – the STEP standard for product data exchange', *Journal of Computing and Information Science in Engineering*, Vol. 1, No. 1, pp.102–103.
- Pratt, M.J. (2005) 'ISO 10303, the STEP standard for product data exchange, and its PLM capabilities', *International Journal of Product Lifecycle Management*, Vol. 1, No. 1, pp.86–94.
- Sauder, D. and Morris, K.C. (1995) 'Design of a C++ software library for implementing EXPRESS: the NIST STEP Class Library', in *Proceedings of EUG '95 – The Fifth EXPRESS Users Group Conference*, Grenoble, France.
- Schenck, D. and Wilson, P. (1994) *Information Modeling the EXPRESS Way*, Oxford University Press, New York.
- SCRA (2006) *STEP Application Handbook*, ISO 10303, version 3 [online]
http://www.uspro.org/documents/STEP_application_hdbk_63006_BF.pdf
(accessed 31 March 2013).
- Tassey, G. (1999) *Interoperability Cost Analysis of the U.S. Automotive Supply Chain – Final Report*, RTI project number 7007-03, Research Triangle Institute [online]
https://www.rti.org/pubs/US_Automotive.pdf (accessed 31 March 2013).
- Tolio, T., Ceglarek, D., ElMaraghy, H.A., Fischer, A., Hu, S.J., Laperrière, L., Newman, S.T. and Váncza, J. (2010) 'SPECIES – co-evolution of products, processes and production systems', *CIRP Annals – Manufacturing Technology*, Vol. 59, No. 2, pp.672–693.
- Valilai, O.F. and Houshmand, M. (2010) 'INFELT STEP: an integrated and interoperable platform for collaborative CAD/CAPP/CAM/CNC machining systems based on STEP standard', *International Journal of Computer Integrated Manufacturing*, Vol. 23, No. 12, pp.1095–1117.
- von Euler-Chelpin, A. (2008) *Information Modelling for the Manufacturing System Life Cycle*, PhD thesis, Department of Production Engineering, Royal Institute of Technology, Stockholm.
- Xu, X.W., Wang, H., Mao, J., Newman, S.T., Kramer, T.R., Proctor, F.M. and Michaloski, J.L. (2005) 'STEP-compliant NC research: the search for intelligent CAD/CAPP/CAM/CNC integration', *International Journal of Production Research*, Vol. 43, No. 17, pp.3703–3743.
- Xu, X.W., Wang, L. and Newman, S.T. (2011) 'Computer-aided process planning – a critical review of recent developments and future trends', *International Journal of Computer Integrated Manufacturing*, Vol. 24, No. 1, pp.1–31.

Paper E

Li, Y., Hedlind, M., Kjellberg, T. and Sivard, G. (2014c). System integration for kinematic data exchange, *International Journal of Computer Integrated Manufacturing*, Vol. 28, No. 1, pp. 87-97.
DOI: 10.1080/0951192X.2014.941937.

System integration for kinematic data exchange

Yujiang Li*, Mikael Hedlind, Torsten Kjellberg and Gunilla Sivard

Production Engineering, KTH Royal Institute of Technology, Stockholm, Sweden

(Received 20 June 2012; accepted 18 June 2014)

Industry needs a system neutral solution for exchange of kinematic models. In this article, the first known valid implementation of kinematic mechanisms based on ISO 10303 Standard for the Exchange of Product (STEP) is presented. The result includes an implementation framework and two developed prototypes. Two major challenges of standard-based development are identified and generalised: data integration and system integration, which are solved by the framework. The two prototypes are implemented to establish kinematic data exchange between Siemens NX® and STEP-NC Machine™ via STEP AP214 files. Experiences of design and development of the applications are presented, and a validated case study of data exchange using the developed applications is shown. There are other attempts of using STEP as basis for modelling, but as the first valid STEP implementation on kinematics, this approach demonstrates the feasibility of pure STEP-based data exchange for kinematic mechanisms. The prototypes also show potential of utilising the framework for general standard implementations. The research is expected to motivate deeper understanding and extensive applications of the STEP standard in industry and academia.

Keywords: kinematics; CAD/CAM; STEP; implementation

1. Introduction

Numerous commercial CAx software systems have been developed and applied in different fields of the Digital Factory. Meanwhile, diverse partnerships have been built between IT software vendors, industrial practitioners, and academic researchers. Issues regarding translation of data formats among diverse software systems become a major barrier for different partners. Therefore, a system neutral solution is required to enable seamless exchange of different types of information, e.g., product geometry, kinematics, tolerances, and classification.

Basic concepts to represent kinematic mechanisms in computer-aided design (CAD) are common among the majority of applications: combinations of links and joints to describe topology and geometry. Different types of motion constraints can be defined, e.g., revolution, translation, and cylinder. The exchange and sharing of kinematic mechanisms is of utmost importance in the field of industrial product data. The well-known standard for product data exchange, ISO 10303 STEP (Standard for the Exchange of Product data), addresses representation of kinematic mechanisms as an integrated application resource, p105 (ISO 10303-105 1996). However, based on this standard, applications for kinematic data exchange are very rare.

STEP is introduced for ‘the representation and unambiguous exchange of computer-interpretable product information’, as stated in ISO 10303-1 (1994). These needs have grown with development of the global market where

organisations collaborate in diverse ways, e.g., virtual enterprises, supply chains, or extended enterprises (Chryssolouris et al. 2008). In this context, it is common that organisations, with different software systems for the same or related functions, have to work together. Therefore, practitioners have to face the complicated problem of seamlessly exchanging and sharing data in a highly collaborative environment. Besides, designers should communicate information not only on product design but also on processes and resources. Thus, standardised data exchange becomes an important solution. The comprehensive structure of the STEP standard is promising to meet this imperative need for neutral product data formats, which makes almost all major CAx software vendors support STEP, especially for representation of three-dimensional (3D) geometry.

However, applications with the support for STEP-based kinematic mechanism data exchange have not yet been widely implemented in industry. At present, most companies have to use slides, fax, telephone, or paper-based documents to describe and exchange kinematic data. Usually skilled CAx operators re-entering the kinematics model is the only option to bridge the gap between different data formats. Such situation is a huge waste of resources.

The widely used STEP application protocol AP214 (ISO 10303-214 2010), using p105, offers a standardised data model schema for the integration of kinematics with geometry and assembly models. Several research projects,

*Corresponding author. Email: yujiang.li@iip.kth.se

as mentioned in Section 2, have tried to implement the kinematic model of AP214. But until now, no known valid implementation of STEP-based data exchange of kinematic mechanism was not found in the literature. Therefore, in Section 3, it is necessary to examine and discuss principles for kinematic modelling for practical implementation. Section 4 introduces an implementation framework that aims at simplifying integrated implementation of standardised data exchange. In Sections 5 and 6, the first known valid STEP-based implementation for kinematic mechanism is presented, together with experiences from the process of design and development. Section 7 presents a case study with the developed applications, which demonstrates the feasibility of kinematic mechanism data exchange based on STEP. Section 8 makes a conclusion of the contribution and a future plan of this research.

2. State of the art in kinematic modelling

In the domain of digital enterprise technology, standardised communication solutions such as STEP are regarded as important technologies by Maropoulos (2003). Kinematic modelling is considered necessary in design, visualisation, and simulation for production modelling and optimisation. With such needs, STEP p105 was published in 1996. However, there has been no known implementation with valid STEP p105 modelling. Instead, almost all found literature regards it as a conceptual model rather than an implementable integrated application resource. An important reason for such a lack is observed that there has been no guide or example on kinematics in any document of the STEP standard. Therefore, the application of STEP-based kinematic modelling only has a simple history, as shown in Figure 1. The events in the timeline of this figure will be introduced in detail in the following part of this section.

This section aims at reviewing research and development of kinematic information modelling regarding standardised data exchange. Related research can be divided

into two groups: schema development and application development. The former focuses on development of national or international standards for kinematics data exchange in different contexts and the latter on application development utilising the standards.

2.1. Schema development

In the beginning of ESPRIT Project 2614/5109 NIRO (Neutral Interfaces for Robotics), the project members proposed a preliminary plan for kinematics in STEP. This proposal was accepted by ISO as a basis for further integration with other parts of STEP (Bey et al. 1994).

Two years later, ESPRIT Project 6457 InterRob (Interoperability of Standards for Robotics in CIME) published a ‘Specification of a STEP Based Reference Model for Exchange of Robotics Models’. The aim of this specification is very similar to application protocols of STEP: accurate exchange of manufacturing data between different systems. Its kinematic schemas were developed mostly based on ISO 10303-105 (1996) (Haenisch et al. 1996). Nevertheless, since it was published, there has been no known application that uses this specification.

MechaSTEP (DIN/PAS 1013 2001) is a German national standard for mechatronic systems. Integrating a large number of STEP parts, it also plays a role as an Application Protocol of STEP but extends the scope of STEP to the area of mechatronics. STEP p105 is reused but modified a little in this standard, so MechaSTEP refers to it as ‘MechaSTEP 10303-105:1996’. Several subtypes of the *sensor_element* for kinematics are added to define the interaction states in the context of multibody systems. This standard provides an example of reusing and extending STEP, but so far there has no known implementation that uses it either.

An expandable conceptual model of assembly information was proposed in a project held by the National Institute of Standard and Technology in USA, named Open Assembly Model (Sudarsan et al. 2005; Rachuri et al. 2006). This model focuses on representation of geometry, kinematics, and tolerances, and it is claimed to use the STEP as the underlying data structure. But it only adopts the concepts defined in p105, rather than the actual data model defined within its schema.

A small subset of p105 was used in a semantics-oriented machine tool modelling approach for 5-axis machining application (Tanaka et al. 2008). The research extended the p105 in its subset model for kinematics. The model was expected to define a machine tool model schema for STEP-NC (ISO 14649).

STEP-NC (ISO 14649) provides an object-oriented data model based on EXPRESS as the next generation of Computer Numerical Control (CNC) program language. It is designed to replace the G code defined in the old standard ISO 6983-1 (2009). The integration with

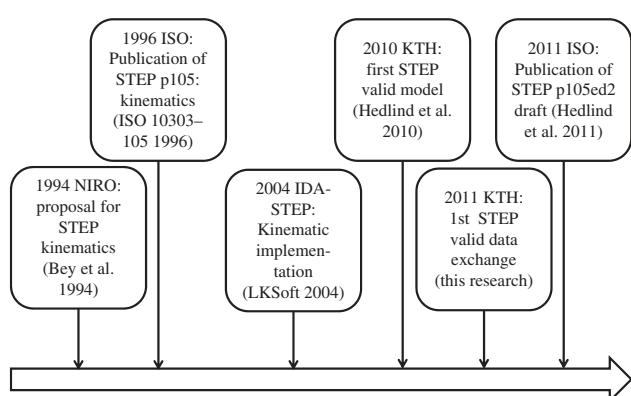


Figure 1. Milestone of kinematic modelling using STEP.

the STEP standard at the schema level enables it to support data exchange with other CAx (Computer-Aided Technology) systems. One example is the draft part ISO/TS 14649-201 (2011) that uses a subset of p105 between the moveable machine tool element and its geometry. Four types of kinematic pairs are selected in this part of STEP-NC: cylindrical pairs, prismatic pairs, revolute pairs, and spherical pairs, as well as support for properties such as pair ranges, pair values, and actuations. The Sheth–Uicker parameters as an alternative for the placement identification are adopted as well. As a new standard published in draft, there has been no known application that implements the kinematics conforming to ISO/TS 14649-201.

STEP-NC compliant Machine tool Data model (STEP-NCMtDm) was developed by Yang and Xu (2008) to implement STEP-NC for process planning. The model plays a role as a database schema for machine tools to enable STEP-NC-based process planning systems. Based on the STEP standard, the model can be presented in EXPRESS (ISO 10303-11 2004) or mapped to eXtensible Markup Language (XML) based on ISO 10303-28 (2007). The kinematic model is a core part of STEP-NCMtDm. Components, pairs, and chains are defined to compose the kinematic structure for movement of tools and workpieces.

Unified Manufacturing Resource Model (UMRM) was presented for various manufacturing resources in the context of CNC machining operations by Vichare, Nassehi, and Newman (2009). EXPRESS was also used to describe the schemas of UMRM. Kinematic aspects were considered as a group of rotation or linear axes owned by all kinds of machine elements in a CNC system. This modelling approach for kinematic mechanism was reused in another research to develop an information model for machine tool process control (Kumar et al. 2010), which was proposed as a compensation of STEP-NC standard. For fixtures, the concepts of joints and links were adopted, but structured in a different way from STEP (Vichare, Nassehi, and Newman 2011). For example, a kinematic joint may be associated with more than two kinematic links and more than one kinematic pair type, referring to the kinematic structure schema of STEP p105.

Automation Markup Language (AutomationML[®]) is a format that partly overlaps the scope of AP214. AutomationML[®] uses COLLaborative Design Activity (COLLADATM) version 1.5 for geometry and kinematics. From the view of usage, COLLADATM is a common data exchange format for digital content creation (DCC) applications in the gaming and animation industry, such as 3Ds Max[®], Maya[®], and XSI[®] (Arnaud and Barnes 2006). Meanwhile, STEP is a widely used and comprehensive international standard for data exchange in the field of manufacturing engineering. Hence, in a research for CAD/CAM systems, STEP is an industrially feasible choice. The utilisation of XML is often mentioned as an

advantage for AutomationML[®] over STEP with the p21 (ISO 10303-21 2002) format. However, XML in STEP has already been published as ISO 10303-28 in 2007, and the mapping from EXPRESS schemas to XML schemas is defined completely. Therefore, the advantages of XML data format in some contexts can benefit STEP as well. COLLADATM added kinematics in 2008 in its version 1.5, which was the version integrated in AutomationML[®]. The added kinematic mechanism is suitable for DCC applications, but several features make it not suitable for manufacturing industry. Only two types of kinematic pairs are defined: revolution and translation. Compound joints have to be used to create pairs with more degrees of freedom (DOFs). As a comparison, various types of kinematic pairs, described in different combinations of DOFs, are defined comprehensively in p105 and will be extended in its second edition.

2.2. Application development

A study about machine control software in the context of industrial economy was an early attempt to involve the kinematic mechanism based on p105 (Birla and Kang 1995). The STEP-based kinematic model was defined for machining processes, in terms of fixtures, workpieces, and tools. But the result did not fully conform to the STEP standard.

An important attempt to implement kinematics of the STEP standard is the IDA-STEP[®] (integrating distributed applications on the basis of STEP data models) project (LKSoft 2004). An outcome of the IDA-STEP[®] project is a software prototype that can access, view, and edit STEP data that can be stored in a STEP database for Internet-based exchange and sharing between multiple devices. This project succeeded to develop ‘an early prototype of a kinematic editor’ and the resulting STEP file has a relatively complete description of the kinematic structure: joints, links, a limited number of pair types, and range values. A Virtual Reality Modelling Language file can also be produced and viewed in a web browser. Still, some errors make it not valid against the standard. According to the report, IDA-STEP[®] as a project is the first implementation attempt based on p105 since it was published.

Another implementation related to p105 that needs to be mentioned is the Space Kinematic Model (SKM) module within the ongoing Thermal Analysis for Space project that aims at building a thermal network and a test environment for space missions (ESA 2007). The SKM module utilises the kinematic structure of AP214 Application Reference Model (ARM) to describe the motion constraints of rigid bodies. But it is the Application Interpreted Model (AIM) schema that is intended for implementation and computer interpretation, rather than the ARM. The usage and mapping relationship of AIM/ARM will be presented in Section 3.

3. STEP kinematic modelling

ISO 10303-105 is a member of integrated application resources of the STEP standard. It provides a data model to support kinematic mechanism exchange for computer-aided kinematic design and analysis. This part of the STEP standard was published in 1996 and then two technical corrigenda were published in 2000. At present, the second edition (STEP p105 ed2) is under development and its first usage will be within the new standard AP242 (Hedlind et al. 2011).

Major features of p105 are topology, motions, and analysis of the kinematic mechanism. Links and joints are used to define the kinematic topological structure. Kinematic pairs define the geometric aspects of the joint and DOFs. These concepts are utilised during design and development in this research.

As an integrated application resource within the architecture of STEP, p105 is not designed to be implemented independently. Its implementation methodology should be defined within the application protocols using it. For implementation of any application protocol, users can start with the scope and information requirements in a set of application activity model (AAM) diagrams presented with Integration DEFinition for Function modelling. At the same time, ARM defines the concepts in terms of application objects and their relationship. In an application protocol, ARM and AAM are connected by Unit of Functionalities (UoFs). The concepts defined in data flow of the AAM are referred to by UoFs, e.g., UoF K1 in AP214 is specified for kinematics representation. The UoFs divide ARM into groups and contain application objects that represent unique concepts in the context of this AP. Readers can use ARM and AAM to understand a conceptual model with a group of connected application objects for information to be modelled. Then, the application objects can be made computer interpretable through ARM/AIM mapping tables. An ordinary STEP file for data exchange is composed of instances structured according to AIM schemas. Application developers should go through the above process and create programs to automate the generation, modification, and access of STEP files in a certain context, e.g., generating STEP AP214 files with kinematics from Siemens NX. During the development, the generated STEP files should be validated against the AP, and the validation result reports errors in terms of constraints and rules, which provide hints for debugging.

STEP AP214 is the only existing application protocol integrating p105. The data model in AP214 can support the kinematic mechanism representation for simulation and data exchange. Topology and motion constraints of the kinematic structure can be defined for both open and closed kinematic chains. Thereby, both serial and parallel mechanisms can be represented. In addition, representation of

motion configuration, control information, and simulation results can be specified to support kinematic simulation.

As common for all parts of the STEP standard, documentation of p105 is comprehensive and detailed. For application developers, it is time consuming to look up and understand all detailed information for each piece of concepts that are distributed in different documents. To reduce workloads for application developers, one of the major deliveries of this research is a solution for integration implementation in two aspects: data integration and system integration, which will be described in Section 4. Besides, the experiences of this research also contribute to the development of the second edition of p105 with an opportunity to reduce the user workloads. This future version will adopt several approaches regarding user-friendly issues: highly integrated data models, friendly documentation technology, and detailed working samples.

4. Research approach

In this research, two types of integration, data and system integration, are identified as major challenges for application developers working with STEP and other standards based on EXPRESS, e.g., ISO 13584, ISO 13399, and ISO 14649. Therefore, the research needs to identify the difficulties of the integrations and develop specific strategies to solve them.

At first, this research meets a challenge to involve kinematic mechanism representation into an existing STEP AP214 file that includes a 3D geometry model, which is the data integration. STEP AP214, as the only published AP integrating p105, retains a great benefit for detailed modelling of process planning. A proper design of data integration can establish seamless association between kinematic mechanism and geometry, providing support for standardised data exchange in different areas of digital factory. An ontology-based approach is applied to support the data integration in the context of the digital factory (Kjellberg et al. 2009), where mapping from concepts to standard models is proposed as a new modelling approach. As a result of this approach, an AP214 valid kinematic model was developed by Hedlind et al. (2010) and is used as a base for development in this research.

This research also provides a solution for implementation of the system integration between extensive STEP translations and existing commercial CAx systems. The STEP standard, regarding geometric representation, has been supported in lots of CAD systems, such as Siemens NX®, AutoCAD®, Pro/Engineer®, and CATIA®. Different types of translators are used in these systems, e.g., independent applications, command lines, or even directly opening/saving actions. Thus, designers and researchers can use STEP files to exchange geometric data of their designs between different software systems. Hence, for

any types of STEP implementation, it is a waste of resources to develop a translator for geometric data models. Integration development reusing native translators is a better choice for development of STEP-based applications such as a kinematic translation integrated with existing CAx systems.

Thus, STEP is treated as a bridge between information systems in manufacturing industry. However, except geometry, many other types of design information are still isolated within different CAx system, such as kinematics, Geometric Dimensioning and Tolerancing (GD&T), and classification. Therefore, integration strategies with existing computer systems are required for extensive STEP implementations. Different interface designs of information systems bring forward different strategies of system integration. For CAx systems, three types of data access interface designs are observed commonly. (1) An Application Program Interface (API) is provided to perform actions within the working session. For instance, developers can use NX Open to build applications integrated within the native interface of Siemens NX®. (2) An API is provided to perform actions with data sets as digital files or databases. For instance, DWG is a common file format widely used in products of Autodesk® and other vendors, for which many commercial or free APIs are provided for data manipulation. STEP is also an example where its interface Standard Data Access Interface (SDAI) ISO 10303-22 (1998) can be bound with common programming languages. (3) Data sets are encoded with common standards such as ASCII, Unicode, or with more specific syntax, e.g., XML, which can be interpreted by information systems. Usually specifications are provided for human reading and modification directly. This research

will provide an implementation framework for such types of integration designs, exemplified as following developments.

The sample software selected to implement the framework are Siemens NX® and STEP-NC Machine™, demonstrating the feasibility to bridge kinematic mechanism between different systems using the STEP standard. The data interface designs of the two systems belong to the first and the third types, respectively, as stated earlier. As a CAD system, Siemens NX® provides functions to define the kinematic model with common objects, e.g., links, joints, motion placements, and motion ranges. For motion calculation, NX® includes two solvers: MSC Adams/Solver® and FunctionBay RecurDyn®. The choice of the STEP-NC Machine™ application is for its ability to simulate tool-paths described in AP238 (ISO 10303-238 2007) together with AP214 machine tool geometry models. However, only geometry models in AP214 files are currently used in STEP-NC Machine™. The kinematic mechanism has to be represented in an XML format of which the schema is natively defined by STEP-NC Machine™.

The system integration design shown in Figure 2 presents the general structure of the framework with the two kinds of example software. The software developed in this research is named KIBOS (KTH Implementation Based On STEP). KIBOS for NX® is executed within the working session of Siemens NX® 7.5. It invokes the native STEP exporter of Siemens NX® to produce a STEP file with only geometry. Meanwhile, it collects kinematic data from the working part in Siemens NX® via NX Open (see Section 5 for technical details). Then, KIBOS for NX® merges the kinematic data with the original STEP file in a

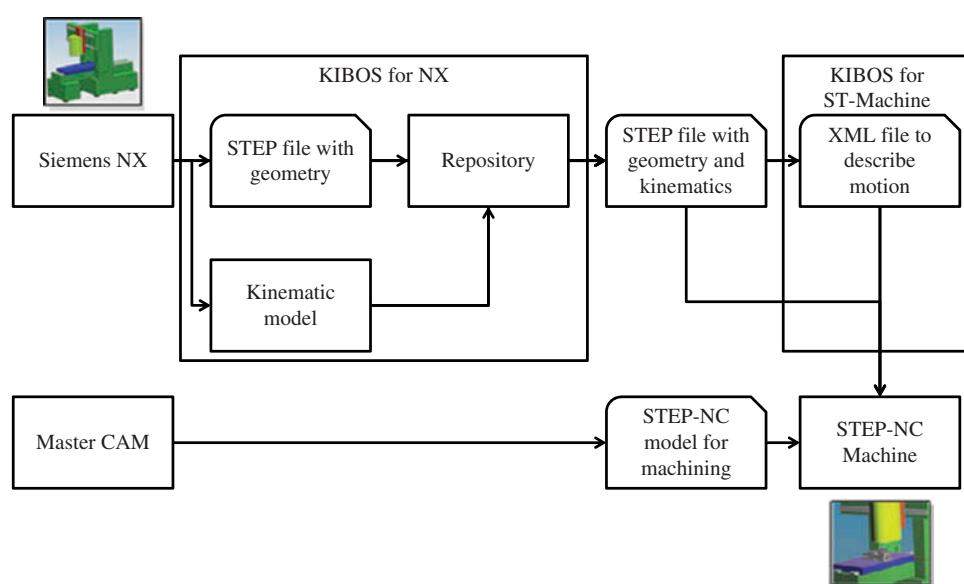


Figure 2. General system integration design.

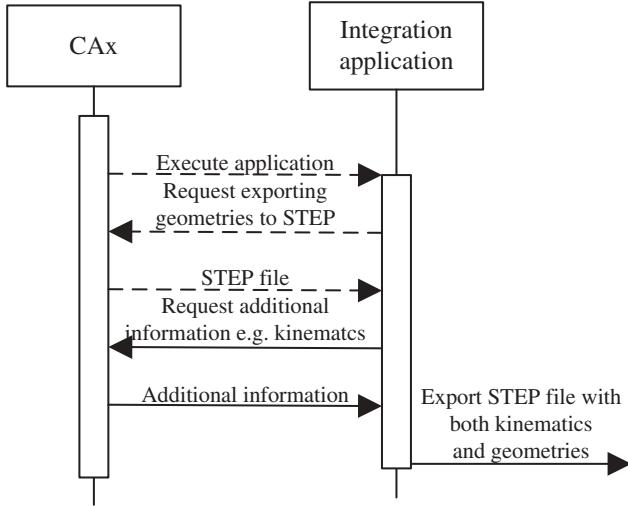


Figure 3. A system integration framework with CAx.

data repository and in the end exports a new STEP AP214 file including both geometry and kinematics.

This integration pattern can be conceptually generalised as the framework shown in Figure 3 for such implementation based on STEP. The framework requires the CAx software with (1) a limited STEP exporting functionality, e.g., geometry, which is common in existing CAx systems; (2) an accessible interface to get the requested information to be exchanged, which is also common in most commercial CAx systems; and (3) optionally, a programming interface to integrate the development within the working session. With the first and second requirements, the development of extensive STEP-based data exchange can be achieved with less effort compared with developing a new translator from scratch. The third requirement is needed only when a highly integrated work flow is needed, which has been tested with Siemens NX® in this research and also with CATIA® in another project by the authors.

The development process of this framework can be roughly divided into two steps: application development and integration development. The application developed in the first step should be able to process both the STEP model and the CAx native model. Then, it merges the information in both models into a new STEP model for further data exchange based on the data integration stated before. The application is in the right part of Figure 3. The second step is to connect the application with the CAx working session or CAx data models, depending on whether the integrated work flow is required. The interaction between the left and right parts in Figure 3 illustrates how the integration should be established. In this example, KIBOS for NX® is designed to be firmly integrated with Siemens NX®. It can be executed within NX® and access NX® functions. Therefore, user's operations can be

simplified largely, compared with independent translators. Such integration can be ignored if it is not necessary or there is no required programming interface. That is why some connections shown in Figure 3 are in dash lines. In that case, the software can be simplified as a translator from one data format to another without the dashed interactions.

On the other hand, STEP-NC Machine™ utilises STEP AP214 for geometry and corresponding XML files for motion description to simulate the machining process with STEP AP238. The XML files conform to a schema defined by STEP-NC Machine™. The direct aim of KIBOS for STEP-NC Machine™ is to make use of kinematics in STEP AP214 for such simulation. It also validates the integration framework shown in Figure 3 in a reverse version, i.e., the kinematic information is extracted and used in the CAx system separately from the AP214 geometry model. On the surface, it is an independent STEP translator. However, the result of the application is not a visible file to be opened by users with CAx, but an added machine tool in the library of STEP-NC Machine™ for motion simulation. The KIBOS reads the STEP file with geometry and kinematics and exports an XML file describing kinematic information to a location for the machine tool library together with the STEP file. This combination can be used to simulate motion of a machining operation in STEP-NC Machine™.

To sum up, extending the STEP-based data exchange in existing CAx systems to other areas is a focus of this research. It can benefit from the integration strategies for this highly integrated environment. An implementation framework is generalised from experiences of the system design and development. With certain requirements, the framework can be easily reused for standardised data exchange in many contexts other than kinematics. Section 5 describes the detailed design of the prototype systems based on this framework, which also demonstrates the feasibility of kinematic mechanism data exchange based on STEP.

5. System design

As stated earlier, this research includes the development of two applications, KIBOS for NX® and KIBOS for STEP-NC Machine™. The two applications are similar in technical background and conceptual design. In this section, they are introduced in detail separately.

Both applications are implementations based on STEP AP214. They are developed with the Java® language, because the relative data sources, i.e., Siemens NX®, XML, and STEP, have powerful programming interfaces for Java®. The API for NX®, named NX Open for Java®, enables access to all functions required in this implementation. Via NX Open, KIBOS for NX® retrieves information from the NX® native models and executes STEP

translations to produce the STEP file with geometry and assembly. ISO 10303-22 SDAI specifies a programming interface to access data models based on EXPRESS, the modelling language for STEP, and some other related information standards. ISO 10303-27 (2000) specifies Java® binding to SDAI and is implemented in JSDAI, an open source development package provided by LKSoftWare® GmbH. Applications in this research are developed with JSDAI. There is no formal schema for the XML defined in STEP-NC Machine™, but the software vendor STEP Tools® offers documented specifications and other technical support for motion description in this file format. There are plenty of methods in Java for XML parsing, of which JDOM is used in this development.

During system design, integration of three layers is focused on data sets, resources, and programs. From the view of user requirements, KIBOS is a program with simple functionality: data format translation. But it needs to be seamlessly integrated with other resources, e.g., NX Open, STEP, STEP-NC Machine™, and JSDAI®, by which the data sets can be communicated. The following sections will describe the integration design of these different parts in detail.

5.1. KIBOS for NX®

The system design of KIBOS for NX® is illustrated in Figure 4. Before development of the application, it is necessary to compile the AP214 AIM schema to an SDAI dictionary with JSDAI®, so that necessary Java® classes can be imported for early binding. At present, it can be expected that AP214 AIM has been stabilised as a published standard, which is the reason to choose early binding rather than late binding. Another reason is that the compiled Java® library by early binding offers the

possibility to validate a part of the program against the schema during compilation. The detailed description and comparison of the two STEP development methods, early and late binding, can be found in Loffredo (1999).

Two programs have been developed representing the two steps of the development process as stated in Section 4. The NX® model processor is based on NX Open API for Java, and the early binding STEP data editor is based on the AP214 library and JSDAI API. The former performs three tasks: (1) processing the NX® model with geometry, assembly, and kinematics; (2) producing a data buffer with kinematics; and (3) invoking the NX® native AP214 translator to output an AP214 file with geometry. Then, the latter merges information in the AP214 file and the data buffer into a new AP214 file with the JSDAI API and the AP214 library.

KIBOS for NX® requires a native NX® model with assembly, geometry, and kinematics. Optionally, in order to make the output STEP file able to be used in KIBOS for STEP-NC Machine™, the user also needs to label the faces where the cutting tool and the workpiece should be placed. In this implementation, the function of Product and Manufacturing Information (PMI) notes in NX® is used to label the surfaces. These notes can be also translated in the same way of kinematic information and stored as general features in the exported AP214 files.

5.2. KIBOS for STEP-NC Machine

The system design of KIBOS for STEP-NC Machine™ (see Figure 5) is similar to KIBOS for NX®. The compiled library of the AP214 AIM schema is also required to read and parse the STEP file by the developed STEP reader. Then, the XML generator creates an XML file according to the data of kinematic mechanism and general features. The XML file is created in a format defined and recognised by STEP-NC Machine™. It stores kinematic information for a certain machine, e.g., kinematic chain definitions, axis definitions, axis placements, and motion

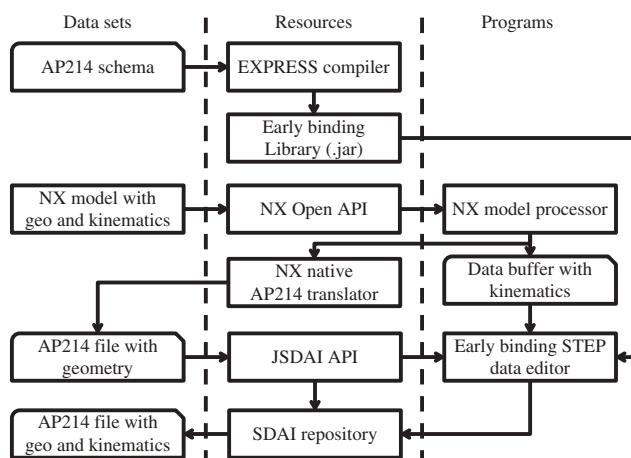


Figure 4. System design of KIBOS for NX®.

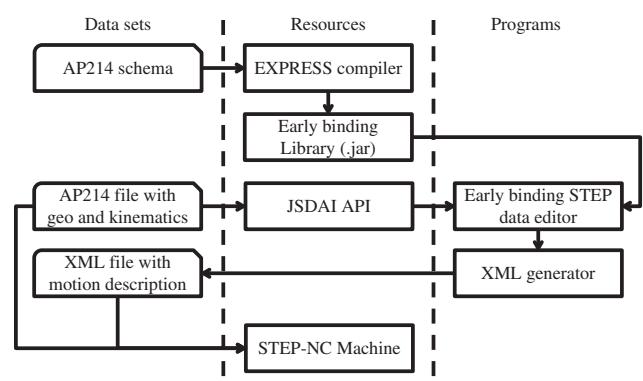


Figure 5. System design of KIBOS for STEP-NC Machine™.

ranges. It also includes placement information for the cutting tool and fixture. KIBOS for STEP-NC MachineTM will automatically place both the STEP file and the XML file in the ‘machine’ folder of the program folder of STEP-NC MachineTM, so that users can use the kinematic information for simulation directly from interface of STEP-NC MachineTM.

6. Implementation

Both KIBOS for NX[®] and for STEP-NC MachineTM are typical implementations of STEP developed in the same development environment. All the development tools used here are open sources or freeware for non-commercial usage. JSDAI[®] and the compiled library of the AP214 AIM schema fully support all operations defined in SDAI and the AP214 data model. The graphical user interface (GUI) of KIBOS is developed with Standard Widget Toolkit, which is an open source library for platform-neutral GUI design and implementation. All the source codes of KIBOS are written and complied with Eclipse[®], which is a well-known open-source Integrated Development Environment.

KIBOS for NX[®] is developed with the focus on high integration with the existing design environment of Siemens NX[®]. The implementation relies on NX Open for Java[®] to interact with the NX[®] session and NX Open MenuScript to integrate user operations. Thus, the application can be used to export the needed STEP file in the same way as other built-in exporters, as shown in Figure 6.

The user interface of KIBOS for NX[®] is shown in Figure 7. It is tailored similar to other common data format translators. The input model is the current working model in NX[®], and the file path and name of the output STEP file can be easily defined in the textboxes of the user interface, or selected from a file dialog by clicking the ‘Browse...’ button.

KIBOS for STEP-NC MachineTM is an independent application without integration of other software. The GUI is designed to finish all necessary configurations, as shown in Figure 8. The user can set the path of STEP-NC MachineTM, the input file, and the output XML file name. The actual orientation of the axis (Z axis) and the reference direction (X axis) of the machine is also able to be defined in case they are not placed in the standard orientations. The kinematic motion solver algorithm can be specified from a list of predefined variants defined by STEP-NC MachineTM. Note that, sometimes the user would not like to repeat typing the same values in the interface. Therefore, a text file is provided together with the program to predefine the default values of all these configurations.

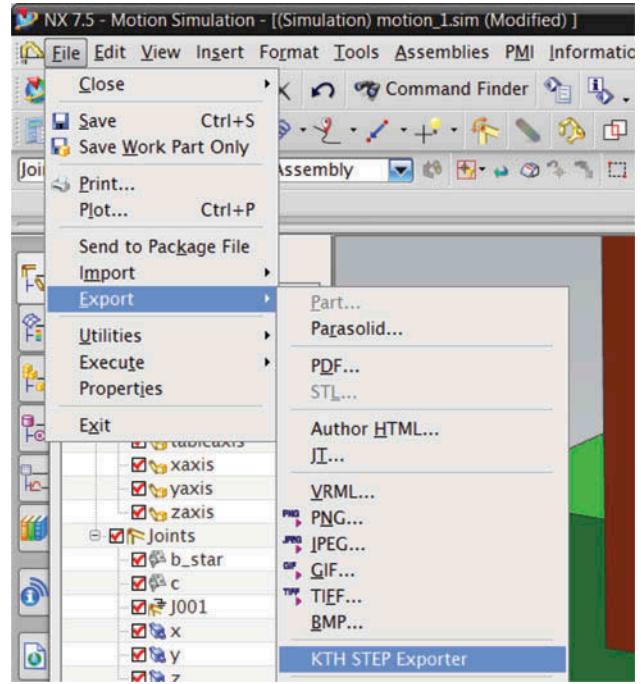


Figure 6. The integrated menu button.



Figure 7. Interface of KIBOS for NX[®].

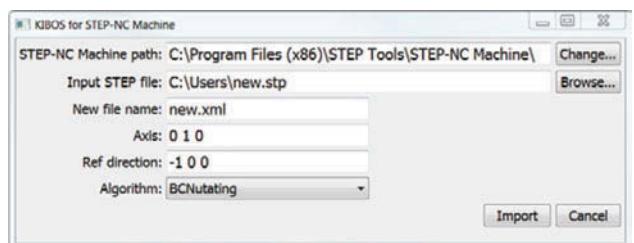


Figure 8. Interface of KIBOS for STEP-NC MachineTM.

7. Case study

This case study demonstrates and validates the presented system neutral solution for kinematic mechanism data exchange, as shown in Figure 9.

A CAD model of a DMG five-axis machine tool is used in this sample. Although it is a simplified model, it still has full capability to demonstrate the motion of its five axes. This sample is used to perform the following tasks:

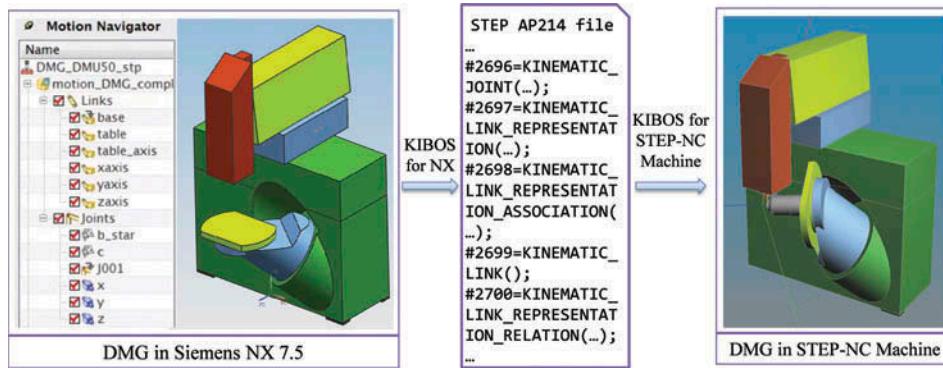


Figure 9. Data exchange between NX[®] and STEP-NC MachineTM.

- (1) creating a kinematic model in Siemens NX[®],
- (2) exporting a STEP file with geometry and kinematics by KIBOS for NX[®],
- (3) importing information within the STEP file by KIBOS for STEP-NC MachineTM, and
- (4) simulating machining operations with the machine tool model in STEP-NC MachineTM.

The first task here is to create a complete kinematic model of this machine with a special kinematic configuration such as axis definitions and motion ranges in Siemens NX[®]. PMI notes are used to label the faces where the tool and the workpiece should be placed. In the component-based motion simulation module of NX[®], six components/subassemblies are selected as the links to form the five kinematic joints to represent the five movement axes of this machine.

Then, using KIBOS for NX[®] and KIBOS for STEP-NC MachineTM, the kinematic mechanism of the machine tool can be imported to STEP-NC MachineTM from Siemens NX[®] via a STEP AP214 p21 file.

The kinematic mechanism is displayed from a motion simulation in STEP-NC MachineTM. An AP238 file for machining of an impeller is used as a sample during this case study. This AP238 file is downloaded from the samples on the official website of STEP-NC MachineTM (STEP Tools Inc 2008). The machining operations described in this file include fixture definitions, tool paths, cutting tools, and operation sequence. During the motion simulation, the five-axis machining can be simulated and displayed in STEP-NC MachineTM. As a small contribution of this research, the DMG machine tool model created here has been now available as a machine tool option in the library provided in the software STEP-NC MachineTM from the version of 9.35 in 2011.

8. Conclusions

This article focuses on the solution of STEP-based integration implementation for kinematic mechanism data exchange with existing commercial CAx software systems. The solution is presented with an implementation framework for system integration development. To validate the solution with a case study, two prototype applications are developed based on the framework. The major features of this solution include the following:

- a seamless system integration with existing CAx systems,
- valid data integration with exported data using CAx native exporter,
- a valid standard model with geometry, assembly, and kinematic mechanism,
- a standardised and economic development environment, and
- a user-friendly interface with existing work flow.

As the first valid implementation for STEP-based kinematic modelling, KIBOS validates the capability of STEP AP214 to represent and exchange kinematic mechanisms. The industrial and academic significance is demonstrated in the presented result of this research. Industrial practitioners are provided with an implementable framework for the standard model exchange between different CAx systems. IT vendors can be benefited by enhancing their products with more comprehensive standardised modelling, which adds value to their current support for standardised geometric modelling. Besides, academic researchers can be assisted by the developed application to create STEP files with the valid kinematic mechanism. In addition to the machine tool motion simulation (as shown in the case study), this result can be applied in other perspectives of the digital factory with the needs of

standardised kinematic modelling, e.g., process planning, machine investment management, factory layout design, manufacturing configuration, and ergonomics. Note that the new standard ISO/TS 14649-201 (2011) uses almost the same data model structure and integration principle for kinematics as STEP. The result of this research also provides a guideline for projects that implement this part of STEP-NC in the future.

This research observes the level of difficulty to implement STEP standard. Many industrial and academic developers have experienced that it can be time-consuming to learn STEP and develop relative applications. A pilot development shows that it takes less than 2 weeks to learn the basic principles and concepts of STEP and how to perform STEP implementation for a Java® beginner with average knowledge on production engineering. One or two months' full-day work is enough to train her or him to become an average STEP application developer. The commercialisation of STEP implementations of geometry has also shown that there is enough implementation potential in industry. In this research, a review of major CAD software systems shows that the geometric data exchange based on STEP AP214 and AP203 have been supported very well by all major vendors: Dassault®, Siemens®, PTC®, Autodesk®, and so on. Note that the solid modelling is one of the most elaborate parts in STEP to understand and implement, and that it has been implemented successfully in different CAD systems. Hence, there are enough development experiences and skilled developers in industry. It can be concluded that the STEP implementation such as kinematic data exchange is feasible for industry if there is strong motivation.

In the current intense development of STEP p105 ed2 and AP242, more resources of the group of this research will be put on the contribution to these new parts of the STEP standard. The test cases and validation of the new standards will be important in the future to make the kinematic data exchange available in industry. It is also essential to establish collaboration between academia, industry, and experienced STEP developers and to enable knowledge transferring between these communities for future research. Hence, during this research, collaborations with several Swedish manufacturing companies start for further case studies in kinematic mechanism modelling and exchange. Multiple CAx systems and development platforms will be involved in future cooperation with industry. The simplification of the implementation methodology is another major issue for populating STEP in industry. Well-designed API is needed to motivate CAx programmers with little or no knowledge of STEP to get into their implementation work rapidly. Besides kinematics, development and validation of standardised modelling approaches in other areas have become future focus in this research project, such as GD&T, classification, and geometry errors (Li, Hedlind, and Kjellberg 2012).

Acknowledgements

We are grateful for the support from Scania®, Sandvik Coromant®, Volvo®, VINNOVA, XPRES (Initiative for excellence in production research), and Siemens® PLM Software, and for fruitful discussions with members of ISO TC184 SC4 WG3 T24.

References

- Arnaud, R., and M. Barnes. 2006. *COLLADA: Sailing the Gulf of 3D Digital Content Creation*. 1st ed. Wellesley, MA: A. K. Peter.
- Bey, I., D. Ball, H. Bruhm, T. Clausen, W. Jakob, O. Knudsen, G. Schlechtendahl, and T. Sørensen. 1994. *Neutral Interfaces in Design, Simulation, and Programming for Robotics*. 1st ed. Berlin: Springer-Verlag.
- Birla, S., and S. Kang. 1995. "Software Engineering of Machine Control Systems: An Approach to Lifecycle Economics." In *Proceedings of 1995 IEEE International Conference on Robotics and Automation International Conference*, edited by Nihon Gakujutsu Kaigi, and IEEE Robotics and Automation Society, 1086–1092. Piscataway, NJ: Robotics and Automation Society, IEEE Service Center.
- Chryssolouris, G., S. Makris, D. Mourtzis, and N. Papakostas. 2008. "Knowledge Management in a Virtual Enterprise - Web Based Systems for Electronic Manufacturing." In *Methods and Tools for Effective Knowledge Life-Cycle-Management*, edited by A. Bernard and S. Tichkiewitch, 107–126. Berlin: Springer-Verlag.
- DIN/PAS 1013. 2001. *Mechastep – Step Data Model for Simulation Data of Mechatronic Systems*. Berlin: Beuth Verlag GmbH.
- ESA (European Space Agency). 2007. "STEP-TAS Technical Details." European Space Agency. Accessed January 28, 2013. http://www.esa.int/TEC/Thermal_control/SEME7NN0LYE_0.html
- Haenisch, J., U. Kroszynski, A. Ludwig, and T. Sørensen. 1996. *Specification of a STEP Based Reference Model for Exchange of Robotics Models: Geometry, Kinematics, Dynamics, Control, and Robotics Specific Data*. 1st ed. Karlsruhe: Forschungszentrum.
- Hedlind, M., L. Klein, Y. Li, and T. Kjellberg. 2011. "Kinematic Structure Representation of Products and Manufacturing Resources." Proceedings of CIRP 7th International Conference on Digital Enterprise Technology, ASTIR Palace Hotel, Athens, September 28–30, 340–347.
- Hedlind, M., M. Lundgren, A. Archenti, T. Kjellberg, and M. Nicolescu. 2010. "Manufacturing Resource Modeling for Model Driven Operation Planning." Paper presented at CIRP 2nd International Conference on Process Machine Interactions, Vancouver, June 10–11.
- ISO 10303-1. 1994. *Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 1: Overview and Fundamental Principles*. Geneva: International Standards Organization (ISO).
- ISO 10303-105. 1996. *Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 105: Integrated Application Resource: Kinematics*. Geneva: International Standards Organization (ISO).
- ISO 10303-11. 2004. *Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 11: Description Methods: the EXPRESS Language Reference Manual*. Geneva: International Standards Organization (ISO).

- ISO 10303-21. 2002. *Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 21: Implementation Methods: Clear Text Encoding of the Exchange Structure*. Geneva: International Standards Organization (ISO).
- ISO 10303-214. 2010. *Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 214: Application Protocol: Core Data for Automotive Mechanical Design Processes*. Geneva: International Standards Organization (ISO).
- ISO 10303-22. 1998. *Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 22: Implementation Methods: Standard Data Access Interface*. Geneva: International Standards Organization (ISO).
- ISO 10303-238. 2007. *Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 238: Application Protocol: Application Interpreted Model for Computerized Numerical Controllers*. Geneva: International Standards Organization (ISO).
- ISO 10303-27. 2000. *Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 27: Implementation Methods: Java TM Programming Language Binding to the Standard Data Access Interface with Internet/Intranet Extensions*. Geneva: International Standards Organization (ISO).
- ISO 10303-28. 2007. *Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 28: Implementation Methods: XML Representations of EXPRESS Schemas and Data, Using XML Schemas*. Geneva: International Standards Organization (ISO).
- ISO 6983-1. 2009. *Automation Systems and Integration – Numerical Control of Machines – Program Format and Definitions of Address Words – Part 1: Data Format for Positioning, Line Motion and Contouring Control Systems*. Geneva: International Standards Organization (ISO).
- ISO/TS 14649-2011. 2011. *Industrial Automation Systems and Integration – Physical Device Control – Data Model for Computerized Numerical Controllers – Part 201: Machine Tool Data for Cutting Processes*. Geneva: International Standards Organization (ISO).
- Kjellberg, T., A. Euler-Chelpin, M. Hedlind, M. Lundgren, G. Sivard, and D. Chen. 2009. “The Machine Tool Model—A Core Part of the Digital Factory.” *CIRP Annals – Manufacturing Technology* 58 (1): 425–428. doi:10.1016/j.cirp.2009.03.035.
- Kumar, S., S. T. Newman, A. Nassehi, P. Vichare, and M. Tiwari. 2010. “An Information Model for Process Control on Machine Tools.” In *Proceedings of the 6th CIRP Sponsored International Conference on Digital Enterprise Technology*, Vol. 66/2010 of *Advances in Intelligent and Soft Computing*, edited by G. Huang, K. Mak, and P. Maropoulos, 1565–1582. Berlin: Springer-Verlag. doi:10.1007/978-3-642-10430-5_118.
- Li, Y., M. Hedlind, and T. Kjellberg. 2012. “Kinematic Error Modeling Based on STEP AP242.” Paper presented at The 1st CIRP Sponsored Conference on Virtual Machining Process Technology, Montreal, May 28–June 1.
- LKSoft. 2004. “Integrating Distributed Applications on the Basis of STEP Data Models, Final Report.” IDA-STEP. Accessed January 28, 2013. <http://www.ida-step.net/sites/ida-step.net/files/IST-2000-30082-Final-Report-v1.pdf>
- Loffredo, D. 1999. “Fundamentals of STEP Implementation.” STEP Tools, Inc. Accessed January 28, 2013. <http://www.step-tools.com/library/fundimpl.pdf>
- Maropoulos, P. 2003. “Digital Enterprise Technology—Defining Perspectives and Research Priorities.” *International Journal of Computer Integrated Manufacturing* 16 (7–8): 467–478. doi:10.1080/0951192031000115787.
- Rachuri, S., Y. Han, S. Foufou, S. Feng, U. Roy, F. Wang, R. Sriram, and K. Lyons. 2006. “A Model for Capturing Product Assembly Information.” *Journal of Computing and Information Science in Engineering* 6 (1): 11–21. doi:10.1115/1.2164451.
- STEP Tools Inc. 2008. “STEP-NC Sample Data: Impeller Part (AP238 file).” STEP Tools Inc. Accessed January 28, 2013. http://www.step-tools.com/products/stepncmachine/samples/impeller/impeller_alternate_1.238
- Sudarsan, R., S. Fenves, R. Sriram, and F. Wang. 2005. “A Product Information Modeling Framework for Product Lifecycle Management.” *Computer-Aided Design* 37 (13): 1399–1411. doi:10.1016/j.cad.2005.02.010.
- Tanaka, F., M. Onosato, T. Kishinami, K. Akama, M. Yamada, T. Kondo, and S. Mistui. 2008. “Modeling and Implementation of Digital Semantic Machining Models for 5-Axis Machining Application.” In *Manufacturing Systems and Technologies for the New Frontier*, edited by M. Mitsuishi, K. Ueda, and F. Kimura, 177–182. London: Springer. doi:10.1007/978-1-84800-267-8_36.
- Vichare, P., A. Nassehi, and S. T. Newman. 2009. “A Unified Manufacturing Resource Model for Representation of Computerized Numerically Controlled Machine Tools.” *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 223 (5): 463–483. doi:10.1243/09544054jem1363.
- Vichare, P., A. Nassehi, and S. T. Newman. 2011. “Unified Representation of Fixtures: Clamping, Locating and Supporting Elements in CNC Manufacture.” *International Journal of Production Research* 49 (16): 5017–5032. doi:10.1080/00207543.2010.518992.
- Yang, W., and X. Xu. 2008. “Modelling Machine Tool Data in Support of STEP-NC Based Manufacturing.” *International Journal of Computer Integrated Manufacturing* 21 (7): 745–763. doi:10.1080/09511920701810691.

Paper F

Li, Y., Hedlind, M., & Kjellberg, T. (2015b). Usability Evaluation of CADCAM: State of the Art, *Procedia CIRP*, Vol. 36, pp. 205-210.
DOI: 10.1016/j.procir.2015.01.053.

CIRP 25th Design Conference Innovative Product Creation

Usability evaluation of CADCAM: state of the art

Yujiang Li^{a*}, Mikael Hedlind^b, Torsten Kjellberg^a^aKTH Royal Institute of Technology, Brinellvägen 8, Stockholm 10044, Sweden^bScania CV AB, Verkstadsvägen, Södertälje 15138, Sweden^{*} Corresponding author. Tel.: +46-8790-8384. E-mail address: yujiang@kth.se**Abstract**

Today, for development of product and production, computer-aided design and manufacturing (CADCAM) maintain a significant role in interaction activities between human and computers. The major objective of computer-aided technology is to simplify engineer's work in collecting, using, and sharing information so that human can maximize the usage of their unique abilities, e.g. creativity and innovation. User experiences (UX) will substantially determine outcomes of such intellectual engineering activities during human computer interaction (HCI) with the CADCAM systems. Usability is a key feature of software ergonomic, and has been standardized as an important property of software quality. Concerns for usability of all kinds of ordinary software interface have been expressed by a lot of studies. Evaluation is always a central activity when practicing usability in an iterative product development process. It can be expected that employment of usability evaluation for CADCAM will be highly valued towards a better level of Human Computer Interaction (HCI). Nevertheless, researches involving usability in this particular domain are very rare. This paper reviews a limited number of publications with such concerns and investigates the current context of use of CADCAM software. Then existing evaluation techniques are introduced and discussed for their feasibility in manufacturing industry based on previous studies.

© 2015 The Authors. Published by Elsevier B.V.

Peer-review under responsibility of the scientific committee of the CIRP 25th Design Conference Innovative Product Creation.

Keywords: CADCAM; HCI; Usability; Usability evaluation; UI**1. Introduction**

Technology makes sense only when it serves human best. Traditional technology has been augmenting human's physical abilities for thousands of years, while information technology will augment human's intellectual abilities, stated by the innovator of the project to build the first computer, Vannevar Bush [1]. No matter how technology evolves, the end users of digital systems will be human, and the ultimate demand of the systems will be satisfying human's requirements. Based on cyber-physical systems and Internet of Things, new concepts such as Industry 4.0 [2], cloud manufacturing [3] and Industrial Internet [4] put high requirements on digitalization of manufacturing industry.

Today, CADCAM systems still remain an important role among the tools for digital manufacturing and digital factory, and take a major part in interaction activities between human and computers in manufacturing industry. This situation will be strengthened further due to evolution towards a high level of flexibility in the manufacturing industry [5]. However,

in academia and industry, it is difficult to find a generally accepted definition regarding CADCAM, and the ones in CIRP Encyclopedia [6,7] are adopted in this article, given its leading authority in this domain:

Computer-aided design (CAD) is the use of a wide range of computer-based tools that assist engineers, architects, and other design professionals in their design activities.

Computer-Aided Manufacturing (CAM) can be defined as the effective utilization of computers in manufacturing [8].

Current implementation of CADCAM mainly concerns functionalities of software systems, i.e. the ability to meet the users' needs. Nevertheless, a complete mixture of functions cannot simply achieve a successful engineering process for product and production development, without concern for how it works to ergonomic experiences of end users. When powerful functions are developed for sophisticated engineering activities, users often encounter troubles to

visualize, understand, and manipulate the functions through designed interaction patterns. CADCAM is supposed to simplify engineers' tasks in collecting, using, creating, and sharing information, but interface designed without consideration of usability often results in unsatisfied experiences and limited outcomes. Such issues can lead to low productivity, time-consuming training and deployment processes, unsatisfied users, and safety risks.

Thus, a way to improve performance of CADCAM systems from the viewpoint of HCI should be explored. Whether an interactive system is usable for human and to which extent the system is usable makes an important property of the system. The property is called usability and exists closely with other terms such as ease of use, user friendliness, user experiences and quality in use in different research areas. ISO 9241-210:2010 [9] defines usability from the perspective of software ergonomics:

Extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use

From the perspective of software product [10], ISO/IEC 25010:2011 [11] adopts this definition of usability as either a subset of quality in use or a main characteristic in its product quality model. Both ISO standards agree that usability is defined as the degree of effectiveness, efficiency, and satisfaction and is affected by the operational context such as users, tasks, environment, technologies, and software interface [12]. The design of interface is merely one aspect of usability analysis [13].

To enable human-centered interactive systems, usability evaluation is a central activity to link other stages of digital product development (e.g. requirements specification, prototyping, design, and implementation) [14]. The fundamental goal of the usability evaluation is to check whether an interactive system is usable, i.e. usability requirements are reached. The evaluation can be performed by system vendors or researchers with optional participation of target users. It can happen through all stages of the product lifecycle of software systems with different purposes. A general procedure includes requirements specification, evaluation planning, conducting, data analysis, and interpretation. A delivery will include usability problems, explanation of the problems, and suggested solutions. Based on usability evaluation, design principles and guidelines can be generated for products in a certain type of context.

Especially for CADCAM, there are a plenty of existing products in use and usability evaluation can help vendors to identify problems and help users in manufacturing industry to select ideal products. It can be imagined that CADCAM brings a challenge of usability evaluation. Several researchers have identified and discussed related problems, which will be the second part of this article. A systematic view to put CADCAM under the microscope of HCI has never been reached. Therefore, the context of use of CADCAM software is analyzed in the third section. Then existing evaluation methods and the feasibility will be discussed based on found literatures in the fourth section.

2. Usability in CADCAM

Compared with other domains extensively concerned by the HCI community (e.g. web design, office productivity software, GPS-based navigation, and mobile applications), research results are very rare related to engineering activities in manufacturing industry and, more specifically, CADCAM. Most of the found literatures were published in 1980s and 1990s, when both areas are amateur and close to each other. Such publications in this highly interdisciplinary topic are usually isolated in different aspects of context, e.g., performance dimensions and user models. Accordingly, a complete picture is difficult to obtain with the limited previous work. Especially for CAM systems, usability may be the least for consideration, with more arguments on functionalities regarding information modeling, system integration, feature recognition, etc. [15] Today, an attempt of clarifying this interdisciplinary topic becomes more challenging when concepts regarding CADCAM and usability are developed deeply for half a century but independently with each other.

Still, it is easy to observe that manufacturing industry has been radically benefiting from the development of HCI. The history of computer-aided technology for engineering activities in manufacturing industry can be dated back to 1960s, when researchers in universities [16] and corporates [17] started to recognize the power of digital computing in aiding human's tasks of design and development. Afterwards, information technology plays a more and more important role in assisting human to perform more types of engineering activities. Meanwhile, the development of computer-related technology in a broader scope has changed our daily life considerably since its birth. The community of HCI devotes decades' efforts to make computers more usable and friendly in this big scope with various crucial contributions that people are taking for granted today, e.g. mouse [18], windows [19], and HyperText [20]. Desktop, mobile, and wearable devices with evolving I/O techniques are changing the ecology of HCI. Indirectly engineers also enjoy these fruitful contributions. With such relevant technologies and interactive styles, CAD is still considered as an important contribution in the history of the HCI society [21].

However, few HCI researchers have paid much attention to CADCAM and other related computer aided technologies in manufacturing industry. The neglect is reasonable due to a relatively small group of users and a considerably high requirement on knowledge compared with other application domains. For projects applying usability evaluation in such a complex domain, Chilana, et al. [22] pointed out that the key challenge was a lack of domain expertise related to the complex systems, which made managers and developers reluctant to trust usability experts. A significant phenomenon is that software vendors tend to trust customers rather than usability experts [23]. Hence, it is recommended for a project with a focus on usability evaluation to collaborate with manufacturing companies in this area.

A limited number of researches presented issues and solutions on the ergonomic perspective of CAD systems. Luczak et al. proposed their opinions on ergonomics in CAD systems in two editions [24,25], which implied a substantial

distinction of engineering activities due to its characterization as mainly in a higher level of cognitive control in informative work, i.e. knowledge-based processes [26]. Such processes may frequently involve problem-solving and decision-making activities which fundamentally differ from rule-based and skill-based processes. A basic demand for the extensive knowledge-based processes is that the design of interaction should be cognitively compatible with users' mental model, e.g. features representing chunks as information clusters to form engineers' knowledge [27]. In the CIRP community, Hatvany and Stone expressed their concerns about user friendliness of CAD systems in 1987 [28]. Most of the proposed principles for CAD were coincidentally included in the Nielsen's heuristics [29,30], but were also specialized for engineering tasks. Visibility of system status, freedom of user controls, flexibility, and efficiency were prioritized in the CAD based research, especially when it came to usage of manuals, sensitivity analysis, and impasses [28].

According to [31], Buxton [32] proposed a five-layer model to describe a classification problem domain of HCI studies: conceptual, semantic, syntactic, lexical, and pragmatic. Traditional usability studies have been usually conducted with the focus on the latter three levels, i.e., syntactic, lexical, and pragmatic, which was attacked by Albers, M. [33] as insufficient for complex information systems where the conceptual and semantic layers dominated the overall performance. When software such as CADCAM is developed and used for complex tasks, usability experts may find that it is convenient to divide a complex system/task into components/subtasks and only investigate them individually. Assumptions have to be made that a set of usable components indicates a usable system. Adequacy of such assumptions has been argued by [34-36]. A high diversity of task paths makes it impossible to achieve a comprehensive understanding or prediction of the interaction patterns [37]. Hence, in the semantic and conceptual layers, Albers [33] required usability studies' focus on reasoning of path selection, predicting unusual scenarios [38], avoiding overloaded information [39], handling mental models [40], and delivering contextual awareness [41].

3. Context of use for CADCAM systems

In the field of HCI, usability should never be treated as an intrinsic property of interactive systems. An entire context of use should be observed, analyzed, and tuned to achieve an acceptable level of usability and, eventually, an acceptable level of user experience [42]. Fig. 1 illustrates basic elements of interaction activities. For end users and other types of stakeholders, the user interface (UI) is equivalent to these types of interactive digital systems [43]. Therefore, interface designers should focus on fitting the interactive system into the context where it is operated [44]. The following part will review relevant studies and discuss the operational characteristics of users, knowledge, tasks, interaction, and environment separately in the particular context of use.

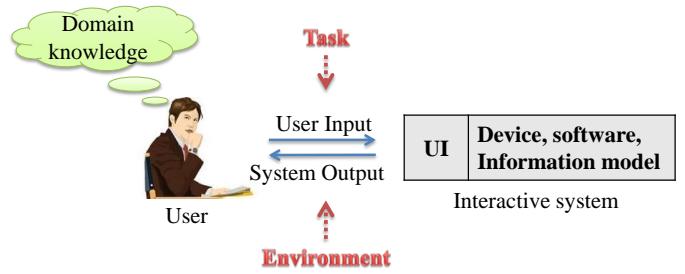


Fig. 1. For the users, UI is the systems (modified based on [43,44])

A major element of this context is users and knowledge they possess, which makes CADCAM significantly different from common software or web applications that usability experts usually handle. In such a complex domain, HCI studies suffer from domain-specific terminology, unique situations, limited access to expertise, reluctance to external professionals, etc. [22] Rasmussen [26] categorized human behaviors into three levels: skill-, rule-, and knowledge-based behaviors, and identified that issues relating to skills and rules were typically in HCI studies' care, while knowledge-based behaviors was a significant research gap for complex domains. The behavior patterns can be easily mapped to a theory of knowledge continuum (Fig. 2) [45,46]. In this continuum, expertise knowledge is most specific, related with professions, and used least frequently. Ideally, lower levels of the continuum should be recommended so that a lower level of users' cognitive loads can be guaranteed during interaction. Meanwhile, usability experts can easily evaluate, analyze and design for any software domain when less specific knowledge dominates interaction. However, during engineering activities, usage of domain-specific expertise knowledge cannot and should not be avoided, especially for convenience of experienced users [47]. It is reasonable because most aids from computers still act similarly to their ancestors (i.e. paper and pen), as recorders for solutions [48]. Major functions of CADCAM coincide with what can be observed by traditional design sketches: external memory, association with non-visual information, and physical settings of the design results [49]. Accordingly, design of IT systems should take into account the traditional cognitive situation of engineers and reach a balance in multiple levels of the knowledge continuum.

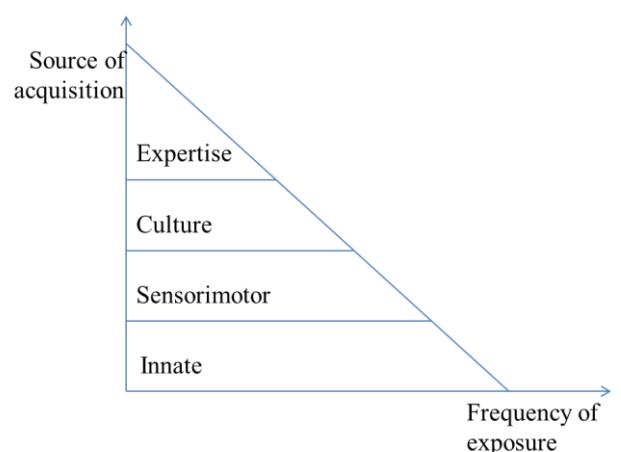


Fig. 2. Continuum of knowledge (modified based on [45])

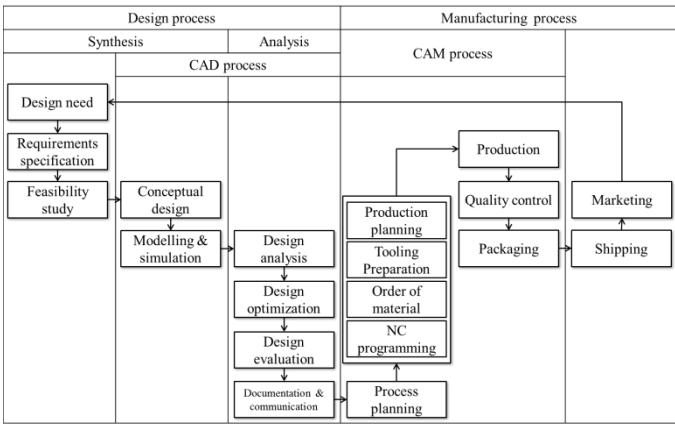


Fig. 3. Tasks performed with CAD/CAM systems (modified based on [7,53])

Today, CAD/CAM software is executing core tasks of information systems for digital factory and digital manufacturing. In academia of manufacturing engineering, many definitions are given for information systems providing digital preparation of products and production, e.g. engineering information systems [50], Computer Integrated Manufacturing (CIM) [51], and virtual product development [52]. Precise unification cannot be reached to clarify the terminology, which leads to a wide diversity when it comes to the categorization of numerous CAx (Computer-Aided technology) software systems, e.g. CAD, CAM, CAE (Computer-Aided Engineering), CAPP (Computer-Aided Process Planning), and CAQ (Computer-Aided Quality assurance). One typical example of the roles of CAD/CAM with brief description of relevant tasks was provided by Zeid [53], which identified a broad scope even including tasks of many other kinds of CAx systems defined by academia (Fig. 3).

No matter with CAD or CAM, interaction has roots in traditional design activities which are a center part of engineering tasks. Psychologists have been studying patterns of design processes for decades. “Problem-solving” cannot simply represent all kinds of related intellectual behaviors of engineers [54]. Moreover, a structured design process cannot guarantee a successful solution, while a flexible methodical style was recommended by experienced mechanical designers [55]. McNeill [56] confirmed a generally accepted engineering design process consisting of a cycle of analysis, synthesis, and evaluation, but the three types of activities are distributed flexibly according to the progression stage. Such behavior classification can be simply mapped to a triple-mode pattern that is critical for novel decision: drawing, examining, and thinking [57]. Rapid shifts between the modes were radical to reach breakthrough decisions [57,58], and therefore require strong supports from software.

The environment, in terms of organizations and technology, may be the least to consider, but still need to be deliberated by designers of CAD/CAM. A phase-oriented task allocation (Fig. 3) is a key driver of CAx software for manufacturing industry where digital technology is required to keep information available, manageable, and integrated [25]. CAD as a center software system for engineering design has been explored thoroughly in terms of e.g. product lifecycle, mass production,

information flows, and enterprise hierarchy [59]. On the other hand, CAM as a bridge between design and manufacturing is also a center of related processes, knowledge, and technology. The both CAx domains should be integrated seamless [60] while also working closely with other technology domains e.g. equipment acquisition and factory design [61].

4. Evaluation techniques and feasibility in CAD/CAM

In research areas of HCI and software engineering, the general requirements of usability are formally documented by the related standards (see Section 1). Based on the standards, Quesenberry’s 5 E’s [62] raised dimensions of usability that should be concerned during evaluation: Effective, Efficient, Engaging, Error tolerant, and Easy to learn. These dimensions guide understanding of requirements and establish a foundation of usability evaluation. Evaluation techniques will be selected, planned, interpreted based quantified or qualified requirements, and finally, guide a design to reach desired performance levels in all the dimensions.

A large number of evaluation techniques for interactive systems have been developed since 1980s. There are various theories on classifications of these techniques, but no one is widely accepted [63]. Most of the classifications are based on [64], according involvement of users and computers (Fig. 4). Development of heuristic evaluation methods [29] and model-based methods (e.g. [65,66]) extends the classification to several new editions [67,68]. Comparisons of usability evaluation techniques [69,70] draw a conclusion which makes the precise classifications not so important: combination of techniques applied at different stages is the best choice. Another conclusion is that proficiency of evaluators in both HCI and task domains is important to identify usability problems of target software [67].

It is a critical decision whether or not to involve participation of users in the evaluation, which leads to two major types of evaluation techniques: user observations and interface inspections. Involvement of users is preferred by most evaluation projects. However, as a part of an iterative product development process, usability evaluation is likely to be performed frequently. Cost of resources and time cannot be afforded if enough users are involved every time. Moreover, before conducting any types of user observations, pilot inspections in-house are recommended to ensure success of formal evaluation sessions.

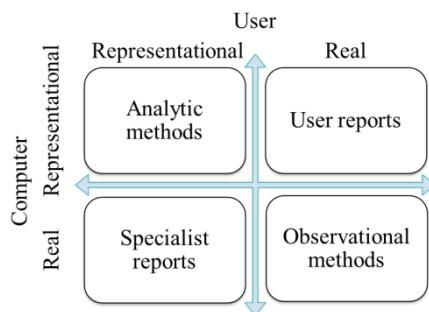


Fig. 4. Classification of evaluation methods [63]

As expected, most of the related studies involve users to evaluate interface and tasks in practice. Ullman [71] directly recorded manual engineering activities to generate seventeen goals of CAD systems. Participants in a collaborative 3D CAD environment were directly observed by Nam and Wright [72], and an additional questionnaire was employed for subjective description of user experiences, which positively supported the proposed collaborative CAD environment by quantified user testing results. In another collaborative context, Ahn et al. [73] used questionnaires as a major means to collect usability data to prove web-based UI outperforming CAD-based tools in accessibility. Sung et al. [48] adopted an indirect observation technique, i.e. logging designer behaviors with CAD systems to achieve knowledge management. The importance of knowledge capture and knowledge capitalization was further supported by Molcho et al. [74], where feedback forms from industry were adopted as a major part of the proposed rule base for manufacturability analysis.

Evaluation techniques without users' participation are also a reasonable choice with advantages such as shorter duration, lower cost, and flexible processing. Heuristic assessment was chosen by Ficarra and Rodríguez [75] for easing learning process of CAD in textile industry. Lee et al. [76] adopted a two-stage evaluation method with a focus on interface inspection by experts. Based on surveys on a large number of novices, experienced experts in usability and 3D engineering were asked to evaluate existing CAD systems and to generate new design principles specified for this domain.

5. Conclusion

Engineers are requested to excel in complex engineering tasks with complex domain knowledge, which makes great difficulties from a novice to an expert [77]. Meanwhile, they have to be skilled with all kinds of functions of CADCAM. Hence, the widely-existing complexity in domain knowledge, tasks, and technology delivers limitations in various dimensions of usability, e.g. efficiency, learnability, and engaging [78]. This situation is compromising user experiences and decreasing productivity in practice.

Concerned with software usability, HCI methodology should be fully explored to improve the performance of CADCAM in manufacturing industry. This paper reviews related publications, analyzes operational context, and discusses existing evaluation techniques in manufacturing industry. Given the trend of digitalization in manufacturing systems, research in this interdisciplinary domain should be carried out with an extensive scope and depth.

Pragmatic issues are also investigated in this paper. Usability studies are usually performed closely with software vendors. In practice, software developers are reluctant to communicate with usability professions, compared with customers' voice. Hence, these types of studies are recommended to be taken by software users. Besides, a high level of proficiency in the domain knowledge is the key to a good result. Moreover, manufacturing companies often have difficulty to select IT products. Evaluation independent of IT vendors will be useful for the customers to select suitable products and for the vendors to address suggestions.

References

- [1] Bush V. As we may think. *The atlantic monthly* 1945; 176(1): 101-108.
- [2] Kagermann H, Wahlster W, Helbig J. Recommendations for implementing the strategic initiative INDUSTRIE 4.0. Technical report 2013.
- [3] Xu X. From cloud computing to cloud manufacturing. *Robotics and computer-integrated manufacturing* 2012; 28(1): 75-86.
- [4] Evans PC, Annunziata M. Industrial internet: Pushing the boundaries of minds and machines. General Electric White Paper 2012.
- [5] Theodorou P, Florou G. Manufacturing strategies and financial performance—The effect of advanced information technology. *CAD/CAM systems*. Omega 2008; 36(1): 107-121.
- [6] Lutters E. Computer-Aided Design. In: Laperrire L, Reinhart G, editors. *CIRP Encyclopedia of Production Engineering*. 2014. p. 252-254.
- [7] Makris S, Mourtzis D, Chryssolouris G. Computer-Aided Manufacturing. In: Laperrire L, Reinhart G, editors. *CIRP Encyclopedia of Production Engineering*. 2014. p. 254-266.
- [8] Groover MP. *Automation production systems and computer-aided manufacturing*. 1st ed. Englewood Cliffs: Prentice-Hall; 1987.
- [9] ISO. ISO 9241-210:2010 Ergonomics of human system interaction-Part 210: Human-centred design for interactive systems. Geneva: International Standardization Organization; 2010.
- [10] Abran A, Khelifi A, Suryn W, Seffah A. Usability meanings and interpretations in ISO standards. *Software Quality Journal* 2003; 11(4): 325-338.
- [11] ISO. ISO/IEC 25010:2011 Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) – System and Software Quality Models. Geneva: International Standardization Organization; 2011.
- [12] Shackel B. Usability-context, framework, definition, design and evaluation. *Human factors for informatics usability* 1991; 21-37.
- [13] Dillon, A. TIMS: A framework for the design of usable electronic text. In: van Oostendorp H and de Mul S, editors. *Cognitive Aspects of Electronic Text Processing*. Norwood: Ablex; 1996. p. 99-120.
- [14] Hix D, Hartson R. *Developing User Interfaces: Ensuring Usability through Product and Process*. New York: Wiley; 1993.
- [15] Xu X, Wang L, Newman ST. Computer-aided process planning—a critical review of recent developments and future trends. *International Journal of Computer Integrated Manufacturing* 2011; 24(1): 1-31.
- [16] Coons SA. An outline of the requirements for a computer-aided design system. In: Proceedings of the May 21-23, 1963, spring joint computer conference. ACM; 1963. p. 299-304.
- [17] Jacks EL. A laboratory for the study of graphical man-machine communication. In: Proceedings of the October 27-29, 1964, fall joint computer conference, part I. ACM; 1964. p. 343-350.
- [18] English WK, Engelbart DC, Berman ML. Display-selection techniques for text manipulation. *Human factors in electronics*, IEEE Transactions on 1967; (1): 5-15.
- [19] Swinehart DC. COPILOT a Multiple Process Approach to Interactive Programming Systems. Technique report. Stanford univ. Calif. dept. of computer science; 1974.
- [20] Nelson TH. Complex information processing: a file structure for the complex, the changing and the indeterminate. In: Proceedings of the 1965 20th national conference. ACM; 1965. p. 84-100.
- [21] Myers BA. A brief history of human-computer interaction technology. *interactions* 1998; 5(2): 44-54.
- [22] Chilana PK, Wobbrock JO, Ko AJ. Understanding usability practices in complex domains. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM; 2010. p. 2337-2346.
- [23] Redish J. Expanding usability testing to evaluate complex systems. *Journal of Usability Studies* 2007; 2(3): 102-111.
- [24] Luczak H, Springer J. Ergonomics of CAD Systems. In: Helander MG, Landauer TK, Prabhu PV, editors. *Handbook of Human-Computer Interaction*. Elsevier; 1997. p. 1349-1394.
- [25] Luczak H, Springer J, Schmidt L. Ergonomics of CAD Systems. In: Karwowski W, editor. *International Encyclopedia of Ergonomics and Human Factors*. 2nd ed. Taylor & Francis; 2006. p. 1088-1093.
- [26] Rasmussen J. Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models. *Systems, Man and Cybernetics, IEEE Transactions on* 1983; (3): 257-266.

- [27] Shah JJ. Parametric and feature-based CAD/CAM: concepts, techniques, and applications. John Wiley & Sons; 1995.
- [28] Hatvany J, Stone BJ. User-friendly CAD systems. *CIRP Annals-Manufacturing Technology* 1987; 36(2): 451-453.
- [29] Nielsen J. Heuristic evaluation. *Usability inspection methods* 1994; 17(1): 25-62.
- [30] Nielsen J. Ten usability heuristics; 2005.
- [31] Foley JD, van Dam A. Fundamentals of interactive computer graphics. Reading: Addison-Wesley; 1982
- [32] Buxton, W. Lexical and pragmatic considerations of input structures. *Computer Graphics* 1983; 17(1): 31-37.
- [33] Albers M. Usability of complex information systems. In: Albers M, Still B, editors. *Usability of complex information systems: Evaluation of user interaction*. CRC press; 2010. p. 3-16.
- [34] Mirel B, Spilka R. *Reshaping technical communication: New directions and challenges for the 21st century*. Routledge; 2002.
- [35] Albers M. Communication of complex information: User goals and information needs for dynamic web information. Routledge; 2004.
- [36] Mirel B. *Interaction design for complex problem solving: Developing useful and usable software*. Morgan Kaufmann; 2004.
- [37] Mirel B. "Applied Constructivism" for User Documentation Alternatives to Conventional Task Orientation. *Journal of business and technical communication* 1998; 12(1): 7-49.
- [38] Sutcliffe A, Karat J, Bødker S, Gaver B. Can we measure quality in design and do we need to? In: *Proceedings of the 6th conference on Designing Interactive systems*. ACM; 2006. p. 119-121.
- [39] Ganzach Y, Schul Y. The influence of quantity of information and goal framing on decision. *Acta psychologica* 1995; 89(1): 23-36.
- [40] Johnson-Laird PN. Mental models: Towards a cognitive science of language, inference, and consciousness. Harvard University Press; 1983.
- [41] Albers MJ. Usability and Information Relationships: Considering Content Relationships and Contextual Awareness When Testing Complex Information. In: Albers M, Still B, editors. *Usability of Complex Information Systems: Evaluation of User Interaction*; 2010. p. 109-131.
- [42] Good M. Seven experiences with contextual field research. *ACM SIGCHI Bulletin* 1989; 20(4): 25-32.
- [43] Stone D, Jarrett C, Woodroffe M, Minocha S. User interface design and evaluation. Morgan Kaufmann; 2005.
- [44] Constantine LL, Lockwood LA. Software for use: a practical guide to the models and methods of usage-centered design. Pearson Education; 1999
- [45] Hurtienne J, Israel JH. Image schemas and their metaphorical extensions: intuitive patterns for tangible interaction. In: *Proceedings of the 1st international conference on Tangible and embedded interaction*. ACM; 2007. p. 127-134.
- [46] Naumann A, Hurtienne J, Israel JH, Mohs C, Kindsmüller MC, Meyer HA, Hußlein S. Intuitive use of user interfaces: defining a vague concept. In: *Engineering psychology and cognitive ergonomics*. Berlin Heidelberg: Springer; 2007. p. 128-136.
- [47] McNamara DS. Reading both high-coherence and low-coherence texts: Effects of text sequence and prior knowledge. *Canadian Journal of Experimental Psychology* 2001; 55(1): 51-62.
- [48] Sung R, Ritchie JM, Rea HJ, Corney J. Automated design knowledge capture and representation in single-user CAD environments. *Journal of Engineering Design* 2011; 22(7): 487-503.
- [49] Suwa, M., Purcell, T., & Gero, J. (1998). Macroscopic analysis of design processes based on a scheme for coding designers' cognitive actions. *Design studies*, 19(4), 455-483.
- [50] Kurbel KE. Manufacturing Systems. In: *Enterprise Resource Planning and Supply Chain Management*. Berlin Heidelberg: Springer; 2013. p. 189-220.
- [51] Radhakrishnan P, Subramanyan S, Raju V. *CAD/CAM/CIM*. New Age International; 2008.
- [52] Hirz M, Dietrich W, Gfrerrer A, Lang J. *Integrated Computer-Aided Design in Automotive Development*. Berlin Heidelberg: Springer; 2013.
- [53] Zeid I. *CAD/CAM theory and practice*. New York: McGraw-Hill; 1991.
- [54] Cross N. Design Cognition: Results From Protocol And Other Empirical Studies Of Design Activity. *Design Knowing and Learning: Cognition in Design Education* 2001; 79.
- [55] Fricke G. Successful individual approaches in engineering design. *Research in Engineering Design* 1996; 8(3): 151-165.
- [56] McNeill T, Gero JS, Warren J. Understanding conceptual electronic design using protocol analysis. *Research in Engineering Design* 1998; 10(3): 129-140.
- [57] Akin Ö, Lin C. Design protocol data and novel design decisions. *Design Studies* 1995; 16(2): 211-236.
- [58] Atman CJ, Chimka JR, Bursic KM, Nachtmann HL. A comparison of freshman and senior engineering design processes. *Design Studies* 1999; 20(2): 131-152.
- [59] Pahl G, Beitz W, Feldhusen J, Grote KH. *Engineering design: a systematic approach*. Springer; 2007.
- [60] Gal-Tzur Z, Shpitalni M, Malkin S. Integrated CAD/CAM system for cams. *CIRP Annals-Manufacturing Technology* 1986; 35(1): 99-102.
- [61] Lundgren M, Hedlund M, Kjellberg T. Process planning methodology guide for interactive learning. In: *Proceedings of SPS2014 The sixth Swedish Production Symposium*. 2014.
- [62] Quesenberry W. The five dimensions of usability. Content and complexity. In: *Information design in technical communication*. 2003. p. 81-102.
- [63] Stowasser S. Methods of software evaluation. In: *The International Encyclopedia of Human Factors and Ergonomics*, 2nd ed. Boca Raton: CRC/Taylor & Francis; 2006.
- [64] Whitefield A, Wilson F, Dowell J. A framework for human factors evaluation. *Behaviour & Information Technology* 1991; 10(1): 65-79.
- [65] Card SK, Moran TP, Newell A. *The psychology of human computer interaction*. Hillsdale: LEA; 1993.
- [66] Bellamy R, John B, Kogan S. Deploying CogTool: integrating quantitative usability assessment into real-world software development. In: *Software Engineering (ICSE) 33rd International Conference*. 2011. p. 691-700.
- [67] Dillon A. The Evaluation of software usability. In: *The International Encyclopedia of Human Factors and Ergonomics*, 2nd ed. Boca Raton: CRC/Taylor & Francis; 2006.
- [68] Cockton G. Usability Evaluation. In: *The Encyclopedia of Human-Computer Interaction*, 2nd ed. Aarhus: The Interaction Design Foundation; 2013.
- [69] John BE, Marks SJ. Tracking the effectiveness of usability evaluation methods. *Behaviour & Information Technology* 1997; 16(4-5): 188-202.
- [70] Andre, TS, Williges RC, Hartson HR. The effectiveness of usability evaluation methods: Determining the appropriate criteria. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 1991; 43(20): 1090-1094.
- [71] Ullman DG. Toward the ideal mechanical engineering design support system. *Research in Engineering Design* 2002; 13(2): 55-64.
- [72] Nam TJ, Wright D. The development and evaluation of Syco3D: a real-time collaborative 3D CAD system. *Design Studies* 2001; 22(6): 557-582.
- [73] Ahn SH, Bharadwaj B, Khalid H, Liou SY, Wright PK. Web-based design and manufacturing systems for automobile components: architectures and usability studies. *International Journal of Computer Integrated Manufacturing* 2002; 15(6): 555-563.
- [74] Molcho G, Zipori Y, Schneor R, Rosen O, Goldstein D, Shpitalni M. Computer aided manufacturability analysis: Closing the knowledge gap between the designer and the manufacturer. *CIRP Annals-Manufacturing Technology* 2008; 57(1): 153-158.
- [75] Ficarra FVC, Rodríguez RA. CAD and Communicability: A System That Improves the Human-Computer Interaction. In: *Human-Computer Interaction. Interacting in Various Application Domains*. Berlin Heidelberg: Springer; 2009. p. 468-477.
- [76] Lee G, Eastman CM, Taunk T, Ho CH. Usability principles and best practices for the user interface design of complex 3D architectural design and engineering tools. *International journal of human-computer studies* 2010; 68(1): 90-104.
- [77] Sweller J. Cognitive load during problem solving: Effects on learning. *Cognitive science* 1988; 12(2): 257-285.
- [78] Kosmadoudi Z, Lim T, Ritchie J, Louchart S, Liu Y, Sung R. Engineering design using game-enhanced CAD: The potential to augment the user experience with game elements. *Computer-Aided Design* 2013; 45(3): 777-795.