

Klaus-Dieter Schewe
Bernhard Thalheim

Design and Development of Web Information Systems

Design and Development of Web Information Systems

Klaus-Dieter Schewe • Bernhard Thalheim

Design and Development of Web Information Systems



Springer

Klaus-Dieter Schewe
UIUC Institute
Zhejiang University
Haining, China

Bernhard Thalheim
Christian-Albrechts-Universität zu Kiel
Institut für Informatik
Kiel, Germany

ISBN 978-3-662-58822-2

ISBN 978-3-662-58824-6 (eBook)

<https://doi.org/10.1007/978-3-662-58824-6>

Library of Congress Control Number: 2019934205

© Springer-Verlag GmbH Germany, part of Springer Nature 2019

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

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

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

This Springer imprint is published by the registered company Springer-Verlag GmbH, DE part of Springer Nature.

The registered company address is: Heidelberger Platz 3, 14197 Berlin, Germany

Preface

The research for this monograph on web information systems started in the mid-1990s, when the second author was confronted with the desire of the local community of his hometown of Cottbus in Germany to design an on-line information service for citizens, tourists and business investors. It would have been easy to exploit available web technology at that time to implement a website for this purpose, but a different approach was taken. On one side this was influenced by the already frustrating experience with the still young web: information one expects to find cannot be found – at least not easily – or one simply gets lost in the abundant mass of information one is not interested to see. How could an elderly citizen be supported by the information service when dealing with the administration, if perusal through websites is already frustrating for experienced scientists? How could tourists or business investors get interested in the region, if they cannot reach quickly the information they need? The conclusion was drawn that the fascinating new technology cannot be the driver for the web-based information service development, but instead the design and development method should start from principal questions such as who would be using the system, for which purpose, with which personal preferences, etc. The key driver for the development must be the needs of the intended users, who are free in their decision to use the system or turn their backs to it.

On the other side, there were still many citizens without computers or access to the internet, so the desire of the community administration was also to reach these people by other communication channels such as the German BTX system that was available in every household with a TV. Of course, it would have been possible to design a separate BTX-based system, but it was recognised that finding a common conceptual abstraction from which different presentations for various communication channels – not even limited to those that were currently available – could be derived would constitute a scientific challenge, the solution of which would have the potential to solve many of the usability problems that were already reported for many web-based information services.

This led to a research programme addressing the design and development of web information systems. Actually, while conducting research in this direction new challenges arose continuously. Nonetheless, around ten years ago the time would have been ripe to integrate all the partial results that had been achieved including the experience from over 30 very large web information systems that had been built and deployed, and to publish all this work in a decent monograph, which could inspire the research community, covering even theoretical foundations, to be used as training material for students and professional web information developers, and of course be applied in more development projects.

As this monograph is only published now, it may appear that it comes late. In particular, the needs of web information systems design and development have significantly changed since the late 1990s. Many books on website development have already been published, and the insight that maintainability of web information systems cannot be obtained by ad-hoc website implementation has led to methods and tools that support database-centred generation of websites and content management systems. Information services, which represent the gist of the so-called web 1.0 systems, were centred around mass information to be produced and delivered by few providers for a large community of users, but the upcoming of web 2.0 systems already changed the picture to a large community of users that also act as providers and the switch from static delivery of information to active communication and process-centric systems. Through web 3.0 and web 4.0 systems the aspect of collaboration and context awareness has been added to create much more challenging questions for web information systems.

Nonetheless or even because of these developments this monograph still comes right in time, as the methodology we developed over years turns out to adequately capture also the challenges that were not yet present at the beginning and thus were neglected by many others, though it would be too bold to claim that we alone addressed them.

So this monograph puts a strong emphasis on the satisfaction of the needs of all stakeholders for whom a web information system is to be built. It also emphasises that these needs can be captured, formalised, integrated in a joint conceptual framework so that an end-to-end, reproducible development process will be enabled. We believe that if systematically applied, our methodology will lead to high quality web information systems, for which many successful practical cases can be used as reference.

Content Overview

This book is structured into four parts and thirteen chapters. Part I covering Chapter 1 is dedicated to a general introduction to web information systems paving the path for the following parts by describing the challenges for web information systems development. Part II covering Chapters 2–6 addresses

methods for high-level design of web information systems, which covers first strategic aspects and second the storyboarding method, which is discussed from syntactic, semantic and also pragmatic perspectives. Part III, which stretches over Chapters 7–10, continues with conceptual design of web information systems including layout and playout. This addresses the decisive web interaction types, the screenography method and adaptation aspects. The final Part IV covering Chapters 11–13 is dedicated to the co-design method for web information system development and its application for the systematic engineering of systems.

In more detail, Chapter 1 introduces the co-design framework for the design and development of web information systems. We first discuss general aspects in conceptual modelling and design, which we then focus on web information systems. This gives us first a general characterisation by six decisive aspects: intention, usage, content, functionality, context and presentation. In many other textbooks in this area the emphasis is usually only on content and presentation with some glimpse of functionality. Second we present the abstraction layer model for web information system development emphasising different layers of abstraction and the dimensions of *focus* (global vs. local aspects) and *modus* (static vs. dynamic aspects).

With Chapter 2 we start the development of the design methodology stressing strategic modelling of web information systems (WISs). This covers first the general characterisation of a WIS by mission statement and brand, utilisation regarded from content, functionality and context angle, and the desired atmosphere, which will impact on layout and playout. Second, the chapter addresses strategic analysis covering linguistic and communication aspects, in particular metaphors used in this context.

Chapter 3 introduces syntax and semantics of the important storyboarding method for WIS. First, story spaces are introduced and formally defined. This leads to scenario modelling, hierarchies of scenes, and plots (aka action schemes) and story algebra, all of which are extended by various detailing aspects and for which different representations are presented. This is complemented by a thorough discussion of actors, their information portfolios, roles, rights and obligations and user profiling. Finally, tasks in a WIS are discussed.

Chapter 4 is marked with ♣ to indicate that it contains content for further, advanced reading. The focus of the chapter is on the customisation of the storyboard to preferences, goals and deontic constraints. For this a formalisation on grounds of Kleene algebras with test is used, which is then exploited for customisation using a conditional term rewriting approach.

Chapter 5 complements Chapter 3 emphasising pragmatics of storyboarding, i.e., what the WIS actually means to its users. After a brief discussion of the role of pragmatics first a method for detailed usage analysis with life cases, user models and actor portfolios is presented. This is then taken further to WIS portfolios centred around information needs and demands leading to content chunks. Finally, an elaborate method for modelling contexts and metaphors used by them is discussed in detail.

Chapter 6 rounds up Part II by showing how the methods for high-level WIS design impact different WIS categories. The chapter discusses the application of the methods in e-business and e-commerce, community and group systems, entertainment and gaming, identity and personal presentation systems, learning and edutainment WIS, and information services.

With Chapter 7 we start the presentation of methods for conceptual WIS design. The chapter introduces web interaction types in a step-by-step way starting with views on conceptual database models, by means of which the separation of global and local content is achieved. There is no fixation of a particular database model, but the query language used for the definition of views must be capable to generate identifiers and links. How this can be done is shown in general. Interaction types result from views by coupling them with operations including a large set of predefined generic operations. In this way the functionality aspect is injected into the conceptual model. Then extensions with respect to adaptivity and granularity are presented, which lead to web interaction types. The former extension permits the splitting or aggregation of information to avoid information overload, whereas the latter extension permits different presentation versions, between which users may switch.

Chapter 8 is again meant for further, advanced reading and thus marked with ♣. It addresses first various extensions of web interaction types with respect to measuring, ordering, presentation options and contexts, then discusses the use of web interaction types beyond content abstraction for single elementary scenes. We discuss how web interaction types can be exploited for session support, navigation contexts and collaboration.

Chapter 9 introduces the screenography method for layout and playout of WIS. It picks up the discussion of the atmosphere in Chapter 2 and discusses the relation of the atmosphere to colouring schemes and grids partitioning the screen. This discussion is taken further to cognitive aspects for visual communication, cognition and design, which finally leads to screenography guidelines and mapping of content and functionality fragments from the conceptual model to the layout and playout design.

Chapter 10 takes the discussion of presentation further emphasising the dependence on culture. For this we start with a discussion of cultural aspects and cultural stereotypes as known from research in psychology. This is then exploited in a method for capturing different cultures in presentations, which we take further down to obtain detailed guidelines for culture-aware storyboards, content and functionality in conceptual WIS models.

With Chapter 11 we start with the WIS development methodology looking first of general principles of co-design, which originates from research in areas such as data-intensive information systems, service-oriented systems, and distributed systems. First the local-as-view approach that is common for the development of data-intensive systems is discussed, which requires dealing with static and dynamic integrity constraints, workflows and view towers. Then co-design of task-centred service-oriented systems is stressed, which adds the decisive aspect of user demand that is so important for WIS. The dual

global-as-view approach that is common for the development of distributed, data-intensive systems is further discussed, by means of which aspects of collaboration enter the general picture. Then all these aspects are merged in an integrated co-design approach for WIS, by means of which all the technical content of the previous chapters is merged into a general methodology for WIS development.

Chapter 12 places the co-design method for WIS development into the context of web engineering. The chapter first discusses the conformity of the methodology with general software engineering quality frameworks such as SPICE and CMMI. It then discusses architecture- and pattern-driven development and illustrates them by taking a glance at the *CottbusNet* WIS and the underlying design and development decisions. This is rounded up by a discussion of WIS development dimensions.

The final Chapter 13 integrates all previous chapters into a method for systematic WIS development, which will take the reader through application domain description, architecture design, requirements analysis, and system specification with the co-design method.

All chapters contain a brief summary of the main facts that should be learnt from the chapter as well as a thorough discussion of relevant literature. To help the user with the orientation within the book a detailed index is added at the end.

Usage of the Book

As indicated above, this book is on one side a research monograph merging almost all our research results in the field of web information systems into a single book. As such it is meant to be used by the research community in this field. However, as we also stressed already that we see the book also as providing training material for students as well as professional WIS developers. Therefore we envision several usages of (parts of) the book for the purpose of further research, teaching and practical development guidance.

With respect to the research community it is of course desirable that the whole content of the book is taken into consideration, although this requires a stretch from fairly formal content (in particular in Chapters 4 and 8, but also partly in Chapters 3 and 7) to content with links to psychology (as in Chapters 9 and 10) or linguistics (partly in Chapters 2 and 5) to content that addresses visual cognition (as e.g., in Chapter 9). This broad spectrum is unavoidable in a field such as web information systems, as will be highlighted in Chapter 1.

Option 1: Use all chapters of the book in sequential order.

However, we admit that not all readers from the scientific community work on all aspects of web information systems, so some chapters can be

omitted without jeopardising the overall understanding of the subject and our methodology.

Option 2: Use chapters of the book in sequential order treating some or all of the following chapters as optional: Chapters 4 and 8 (advanced foundations), Chapter 6 (high-level WIS modelling for various system categories), Chapter 10 (adaptation of presentation to culture).

For teaching courses for students in addition to these two options some clusters of chapters may be selected to emphasise a particular subject in WIS design and development. For instance, courses might want to emphasise “Conceptual Modelling of WIS” (Option 3), “WIS Foundations and Reasoning” (Option 4), “Co-Design and WIS Engineering” (Option 5), or “WIS Layout and Playout Development” (Option 6).

Option 3: Conceptual Modelling of WIS. Use Chapters 1, 2, 3, 5 and 7 of the book in sequential order.

Option 4: WIS Foundations and Reasoning. Use Chapters 1, 3, 4, 7 and 8 of the book in sequential order.

Option 5: Co-Design and WIS Engineering. Use Chapters 1, 2, 3, 5, 6, 11, 12 and 13 of the book in sequential order.

Option 6: WIS Layout and Playout Development. Use Chapters 1, 2, 9, 10 and 13 of the book in sequential order.

All these options can be used by professional WIS developers as well.

Acknowledgement

Many people contributed to the research and the applications we present in this monograph. We would like to thank them all for the willingness to collaborate with us and for the valuable insights they provided through discussions, joint research as well as Master and Ph.D. theses.

Naturally, our teams at Brandenburg University of Technology in Cottbus, Christian-Albrechts-University Kiel, Massey University in Palmerston North and Wellington and Software Competence Center Hagenberg provided most valuable direct contributions to this work and its realisation in

large websites. In particular, we are grateful to our colleagues Gabriele Bogacz, Edith Buchholz, Claus Fellbaum, Roland Kaschek, Felix Kossak, Claire Matthews, Hans-Georg Meißner, Ivor Nissen, Catherine Wallace, Susan Yigitbasi, and Thomas Ziebermayr, to our (former) Ph.D. students Margita Altus, Markus M. Berg, Alexander Bienemann, Antje Raab-Düsterhoff, Thomas Feyer, Kai Jannaschk, Steffen Jurk, Markus Kirchberg, Meike Klettke, Frank F. Kramer, Jana Lewerenz, Hui Ma, Thomas Moritz, Christine Natschläger-Carpella, Istvan-Tibor Nebel, René Noack, Srinath Srinivasa, Martin Steeg, Marina Tropmann-Frick, Qing Wang, Vojtech Vestenicky, and Jane Zhao, to our (former) students Cornell Binder, Annette Borchert, Kerstin Buchholz, Wolfram Clauß, Haiko Cyriaks, Stefan Dieringer, Lutz Feichtinger, Gunnar Fiedler, Christian Galke, Thomas Gutacker, Birger Hein, Birk Heinze, Fynn Holst, Christopher Horn, Holger Kache, Jost and Veit Kannegießer, Zalan Kramer, Udo Krautz, Andreas Krohn, Thomas Kuss, Thomas Kobienia, Felix von Lehn, Sophie Liermann, Holger Mehlan, Thomas Mielke, Günter Millahn, Jana-Cordelia Petzold, Michael Radigk, Andreas Renk, Manfred Roll, Irina Romalis, Oleg Rostanin, Thomas Raak, Sabine Radochla, Faizal Riazud-Din, Steffen Sämann, Peter Schliwe, Michael Schmidt, Peggy Schmidt, Thomas Schmidt, Sven Schoradt, Julia Sonnberger, Thomas Schwanzara-Benoit, Rene Schwietzke, Kati Selig, Bernd Tschiedel, Thomas Voigt, Jens Wölkerling, and Sergiy Zlatkin, and to our company partners in website development projects.

We like to thank Ajantha Dahanayake for the many discussions on modeling issues, Klaus-Peter Jantke and his team for the collaboration in the area of e-learning, which led to the DaMIT system, Sabah Al-Fedaghi and Joachim Biskup for the insights on privacy that we gained from discussions with them, Hannu Jaakkola and his team for helping us to understand the impact of culture and the SPICE quality framework, and Paul Johannesson and his team for the partly controversial discussions on services.

We are further thankful to colleagues from the conceptual modelling community. In particular, we express our thanks to Yasushi Kiyoki, David Embley, Heinrich Mayr, Stephen Liddle, Oscar Pastor and Veda Storey for the many discussions on conceptual modelling of web information systems.

Last but not least we are grateful to Noam Chomsky for the inspirations by his work on governance and binding and the discussions we had with him about its use in web information systems design, and to Egon Börger for the many discussions on the collaboration on Abstract State Machines and their application in business process modelling, which both provided stimuli for this monograph.

Contents

Part I Web Information Systems – General Aspects

1	The Co-Design Framework	3
1.1	Conceptual Modelling and Design	4
1.1.1	Modelling Web Information Systems	4
1.1.2	Context Modelling	6
1.1.3	Large, Distributed and Cooperative Systems	7
1.1.4	Categories of Web Information Systems	8
1.2	Characteristics of Web Information Systems	10
1.2.1	Intention	10
1.2.2	Usage	13
1.2.3	Content	15
1.2.4	Functionality	16
1.2.5	Context	18
1.2.6	Presentation	19
1.3	Abstraction Layers in WIS Modelling and Design	20
1.3.1	Strategic Layer	21
1.3.2	Business Layer	22
1.3.3	Conceptual Layer	24
1.3.4	Presentation Layer	26
1.3.5	Implementation Layer	26
1.4	Bibliographical Remarks	27
1.4.1	General Literature on Web Information Systems	27
1.4.2	The Co-Design Framework	27

**Part II High Level WIS Design – Strategic Analysis and Usage
Modelling with Storyboarding**

2 Strategic WIS Modelling	33
2.1 General Characterisation of a WIS.....	34
2.1.1 Mission Statement and Brand.....	34
2.1.2 Utilisation Space	37
2.1.3 Utilisation Portfolio	40
2.1.4 Utilisation Context	42
2.1.5 The Atmosphere of a WIS	44
2.2 Strategic Analysis	46
2.2.1 Linguistic Analysis Using Word Fields	46
2.2.2 Communication Analysis	50
2.2.3 Metaphors.....	53
2.3 Bibliographical Remarks	57
3 Storyboarding	61
3.1 Story Spaces	62
3.1.1 Scenario Modelling	62
3.1.2 Examples in E-Business and E-Learning	66
3.1.3 Adding Details to Actions and Scenes	68
3.1.4 Hierarchies of Scenes.....	71
3.1.5 Plots and Story Algebras	75
3.1.6 Examples of Plots	80
3.1.7 Alternative Representations for Scenarios and Plots....	83
3.2 Actor Modelling	91
3.2.1 Information Portfolios.....	92
3.2.2 Roles, Rights and Obligations	93
3.2.3 User Profiles and Types	96
3.3 Task Modelling	103
3.3.1 Tasks and Subtasks	103
3.3.2 Representation Means for Tasks	104
3.4 The Complete View of Storyboards	106
3.5 Bibliographical Remarks	107
4 Semantics and Inferences on Storyboarding ♣	111
4.1 Story Algebra and Personalisation	111
4.1.1 Formalisation of Scenarios and Plots	112
4.1.2 Customisation with Respect to Preferences and Goals ..	115
4.1.3 Conditional Term Rewriting on KATs	118
4.1.4 Church Rosser Property	124
4.2 Compatibility of Preference Rules with Deontic Constraints ..	128
4.3 Bibliographical Remarks	131

5	Pragmatics of Storyboarding	133
5.1	The Role of Pragmatics	134
5.1.1	Conceptual Structures in Web Information Systems	135
5.1.2	Information Versus Content	136
5.2	Usage Analysis	137
5.2.1	Facets of Intention	138
5.2.2	Life Cases	144
5.2.3	User Models	152
5.2.4	Actor Portfolios	158
5.3	WIS Portfolios	169
5.3.1	Information Need and Demand	170
5.3.2	The Concept of Persona	171
5.3.3	Content-Centred Analysis	174
5.3.4	Content Chunks for the Entry Scene	177
5.3.5	Story Portfolios	179
5.4	Contexts and Metaphors	180
5.4.1	Contexts of Web Information Systems	181
5.4.2	Towards Context Theory	190
5.4.3	The Metaphor Concept for Web Information Systems	193
5.4.4	Application of Metaphors in Storyboarding	195
5.5	Bibliographical Remarks	198
6	Categories of Web Information Systems	201
6.1	E-Business and E-Commerce	201
6.1.1	Branding	202
6.1.2	Actor Specification	203
6.1.3	Action Verb Fields and Scenarios	204
6.1.4	Elicitation Strategy	205
6.1.5	Supporting Features	209
6.2	Communities and Groups	212
6.2.1	Branding	213
6.2.2	Actor Specification	214
6.2.3	Verb Fields, Functionality and Scenarios	216
6.2.4	Content Chunks	221
6.3	Entertainment and Gaming Systems	222
6.4	Identity and Personal Presentation	225
6.4.1	Branding	225
6.4.2	Word Fields and Scenarios	231
6.4.3	Adaptation	234
6.5	Learning and Edutainment	235
6.5.1	Branding	236
6.5.2	Word Fields and Learning Scenarios	237
6.5.3	Supporting Features	242
6.6	Information Services and Infotainment	244
6.6.1	Storyboard Development	245

6.6.2	System Organisation	249
6.6.3	Life Cases and Derived Functionality	250
6.7	Bibliographical Remarks	254

Part III Conceptual WIS Design – Rigorous Modelling of Web Information Systems and Their Layout with Web Interaction Types and Screenography

7	Web Interaction Types	259
7.1	Interaction Types	261
7.1.1	Capturing Information Consumption	262
7.1.2	Coupling with Databases	263
7.1.3	Entity-Relationship-Based Interaction Types	268
7.1.4	Operations on Interaction Types	273
7.1.5	Alternative Form-Based Approaches	280
7.2	Adaptivity	284
7.2.1	Cohesion Preorders	285
7.2.2	Proximity Values	287
7.2.3	Adaptation of Operations	288
7.2.4	Adaptivity by Means of Aggregation Operations	289
7.2.5	Adaptivity Extension	291
7.3	Granularity	291
7.3.1	Hierarchical Versions	291
7.3.2	Adaptation of Operations	293
7.4	Web Interaction Schemata	294
7.5	Bibliographical Remarks	295
8	Advanced Web Interaction Concepts ♣	299
8.1	Extensions to Web Interaction Types	299
8.1.1	Measuring Systems and Ordering	299
8.1.2	Presentation Options	301
8.1.3	Contexts	303
8.1.4	Extended Web Interaction Schemata	305
8.2	Session Support, Navigation Contexts and Collaboration	306
8.2.1	Web Interaction Types Associated with a Session	307
8.2.2	Context Injection	308
8.2.3	Collaboration in Web Information Systems	309
8.3	Bibliographical Remarks	312
9	Screenography	313
9.1	Development Prerequisites	314
9.2	Elements of Screenography	316
9.2.1	Atmosphere	316
9.2.2	Atmospheric Effect of Colour Schemes	317

9.2.3	Layout Patterns	318
9.2.4	Grid Geometry	319
9.3	Cognitive Aspects of Screenography for Layout	320
9.3.1	Principles of Visual Communication	321
9.3.2	Principles of Visual Cognition	322
9.3.3	Principles of Visual Design	323
9.4	Application of Screenography: Case Study	324
9.5	Screenography Guidelines and Frames	325
9.5.1	Web Page Pattern	327
9.5.2	Web Page Grids	329
9.6	Bibliographical Remarks	333
10	Adaptation of Presentation to Culture	335
10.1	Understanding Cultural Differences	336
10.1.1	The Layered Structure of Culture	336
10.1.2	Kinds of Culture	337
10.2	Cultural Stereotypes	338
10.2.1	The Hofstede Model of Cultures	339
10.2.2	The Lewis Model of Cultures	339
10.2.3	Multidimensional Aspects of Culture	341
10.2.4	Cultural Stereotypes and WIS	344
10.3	Presentation Cultures	346
10.3.1	Deriving Guidelines for Presentation from Stereotypes .	346
10.3.2	Cultural Stereotypes, User Models and Information System Design	347
10.3.3	Cultural Stereotypes and Their Utilisation for System Development	351
10.4	Technologies for Realisation of Culture-Aware Systems .	353
10.4.1	Culture-Aware Storyboards	354
10.4.2	Culture-Aware Content	356
10.4.3	Culture-Aware Functionality: Search	359
10.5	Bibliographical Remarks	373

Part IV Rationale of the Co-Design Methodology and Systematic Development of Web Information Systems

11	The Co-Design Methodology	377
11.1	Co-Design of Schema-Centric Database Systems: The Local-as-View Approach	379
11.1.1	Static Integrity Constraints	382
11.1.2	Representation Alternatives	383
11.1.3	Dynamic Integrity Constraints	384
11.1.4	Specification of Workflows	385
11.1.5	View Towers for Information Systems	386

11.2 Co-Design of Socio-Technical Systems: Database Services in the Task-Centred Approach	398
11.2.1 Concerns for Web Information Systems.....	401
11.2.2 Application- and User-Driven Design of Systems	403
11.2.3 Services that Satisfy the User Demand	404
11.2.4 Task-Centred Development for Database Systems as a Service	405
11.2.5 Database and Knowledge Base Systems that Support Services	405
11.3 Co-Design of Distributed Database Systems: The Global-as-View Approach to Collaboration.....	407
11.3.1 Collaboration of Distributed Systems	407
11.3.2 Architectures for Distribution	408
11.3.3 Coordination Specification and Contracts.....	409
11.3.4 Exchange Frames for Distribution	410
11.4 The Story Space in the Co-Design Approach	410
11.4.1 The Story Space	411
11.4.2 Natural Language Dialogues	412
11.4.3 Web Interaction Types for Information-Intensive Systems	415
11.4.4 The Onion Approach to Website Realisation	415
11.5 Transformation of Web Information Systems	417
11.5.1 Mapping of the Website Specification to Business Layer Models	420
11.5.2 Web Page Extraction	420
11.5.3 Configuration of the Web Page to User Context by Containers	421
11.6 Bibliographical Remarks	424
12 Web Information Systems Engineering.....	429
12.1 The SPICE Methodology to Development of WIS.....	430
12.1.1 Application Domain Description, Requirements Prescription and Systems Specification	430
12.1.2 Dimensions for WIS Engineering	436
12.1.3 Work Products of WIS Engineering	447
12.1.4 The Relationship of WIS Engineering to SPICE	451
12.1.5 Orchestration of WIS Development for Managed Engineering.....	452
12.1.6 Evolving the Co-Design Framework by SPICE	453
12.2 Architectures of Web Information Systems	457
12.2.1 An Architectural Framework.....	457
12.2.2 Architecture-Driven Development	460
12.2.3 Pattern-Based Development	463
12.2.4 Architecture Blueprint	463
12.2.5 The <i>CottbusNet</i> Design and Development Decisions	463

12.3 WIS Development Dimensions	465
12.3.1 Primary WIS Development Dimensions	465
12.3.2 Secondary WIS Development Dimensions	465
12.3.3 The Quality of WIS	466
12.3.4 The Semiotics Background and Pragmatism	468
12.4 Bibliographical Remarks	469
13 Systematic Development of Web Information Systems	471
13.1 Application Domain Description	472
13.1.1 Application Domain Description and Requirements Statement	472
13.1.2 Contracting and Documenting	481
13.2 Architecture Design	487
13.2.1 Architectures and WIS Development	487
13.2.2 Architecture-Driven Engineering	492
13.3 Requirements Analysis	497
13.3.1 WIS Requirements Analysis	497
13.3.2 Companion Activities at Requirements Prescription Layer	506
13.4 Presentation System Specification	508
13.5 WIS Specification (Design and Development)	516
13.6 Bibliographical Remarks	526
13.6.1 WIS Development	526
13.6.2 Realisation of Web Information Systems	529
Bibliography	531
Index	581

Part I

Web Information Systems – General Aspects



The Co-Design Framework

The goal of this chapter is to give a general introduction to the subject of this book, i.e., the design and development of Web Information Systems. In general, every data-intensive information system that is realised in a way that users can access it via web browsers will be called a *web information system* (WIS). This suggests of course that web information systems are just a particular subclass of information systems, and the immediate question is how to characterise this specific subclass.

However, if we understand an information system as a representation of a section of reality in a computerised system that is used for supporting some processes in business, science or engineering, then this information system can be regarded as a “model” of that section of reality. Furthermore, the purpose of the system is that it is to be used by someone for a particular task. Then the major difference in a web information system is that the users, their intentions and their tasks are not known. As a consequence, a web information system is not just a specific information system; it adds a new quality to information systems. Therefore, it is important not only to understand the specific characteristics of web information systems, but also to be able to handle the added functionality.

We start this chapter with a general section on modelling and design, which is intended to highlight the problems in WIS modelling in general, and to draw the fine line of distinction between the modelling needs for web information systems and those for traditional enterprise information systems.

We continue with a discussion of requirements for web information systems. Many authors focus mainly on the realisation of web information systems by web sites, and develop criteria for well-designed sites. Some of these criteria are without any doubt useful, others are at least debatable. The crucial point, however, is that the criteria focus on the presentation of pages, whereas the most critical problem in modelling and design is to manage the complexity of the complete system. Therefore, we draw attention to the requirements arising from large, distributed and co-operative systems. This discussion leads to a categorisation of web information systems. Some of these categories such

as electronic business systems or information services will be used later to illustrate examples.

We continue with a discussion of characteristic aspects of web information systems, i.e., we derive the basic questions that will guide the design and development process. Each of the characteristics will be illustrated for some of the web information system categories.

Finally, we present a framework for the methodology for the design and development of web information systems based on an *Abstraction Layer Model*. Details of this methodology will be filled in the other parts of the book.

1.1 Conceptual Modelling and Design

Information Systems in general and Web Information Systems in particular combine aspects of structure, functionality and interaction. In order to model and design such systems we need a modelling theory that supports all these aspects in an integrated way. Unfortunately, the literature on information systems does not yet contain a complete theory of information systems modelling. Only fragments covering the theory of databases, the theory of processes, or the theory of workflows exist. In the following we will bring some of these fragments together without intending to develop a full theory of modelling.

1.1.1 Modelling Web Information Systems

Modelling in general means the creation, modification, analysis and usage of abstract *models* of a section of reality. Thus, the notion of “model” is based on several features:

mapping: A model represents a section of reality, i.e., there is a mapping from this section of reality to the model.

abstraction: A model is abstract, i.e., it uses unambiguous formal terms the meaning of which is not given a priori, but has to be defined.

reduction: Only features that are considered to be “relevant” and “important” will be represented in a model.

pragmatics: A model depends on the application context, the time and the decisions of the modelling individuals. It may be changed.

Thus, during modelling a lot of decisions will be made. These decisions concern the representation of reality in the model (syntax), the meaning of this representation (semantics), and the distinction between relevant and irrelevant aspects (pragmatics). These decisions are influenced by assumptions made by the modeller, answers to questions asked by the modeller, if this possibility exists, and the interpretation of the available information by the modeller.

There is a useful *pons asinorum* that helps to ask the right questions and make these decisions. A modeller should ask:

- Who will be using the system?
- When will the system be used?
- Where is the information system used?
- What is represented in the system?
- How will the system be used?
- Why is the system used?

As all these questions contain the letter W in the question particle, this approach is sometimes referred to the “principle of the six big W”.

In the tradition of information systems it was usually assumed that systems are used inside some enterprise to support business and working processes. Under this assumption the question who will be using the system is usually answered easily. The users are those people, usually employees, who will work with the system and use it as a tool to do their work. As a consequence, the answer to this question usually leads to workflow and user interface models reflecting various perspectives on the system as defined by the users, by management, and by the modeller.

The question what is represented in the information system refers to the data content. It is usually related to the modelling and design of a database, in which this data will be stored. The criteria that determine the model are influenced by the fact that data must be stored persistently, and made concurrently accessible to multiple users without violating consistency or integrity. In addition, it also concerns the external views, i.e., the data that are made available to a user for a particular task. That is, this question mainly concerns the static aspects of the system.

In enterprise information systems the question when the system will be used is closely related to the questions how the system will be used and why it will be used. These questions lead to the definition of operations that support the business processes. These processes are usually triggered by the advent of certain business cases. Thus, we may say that these questions mainly concern the dynamic aspects of the system and the system usage.

Similarly, the question where the information system is used is considered to be of lower importance, as enterprise information systems are usually used in-house, i.e., within the boundaries of a particular enterprise.

In summary, the six big W-questions basically lead to determining the static and dynamic aspects of the system, and the interaction with its users.

Access to an information system via the web adds a new quality, as the answer to the six questions above is no longer as easy as before. Still the question what is represented in the web information system refers to the data content, and therefore is still related to the modelling and design of a database. Similarly, the question how the system will be used leads to the definition of operations that support certain processes. That is, we preserve the fact that modelling a web information system has to deal with static structures and dynamic functionality.

However, the answers to the other four questions differ significantly from the answers we sketched for traditional enterprise information systems. Starting with the question who will be using the system, the answer is that the users are not known in advance. In other words, web information systems are open to almost everyone who would like to use them. This implies that modelling web information systems has to identify the target users.

Similarly, it is not clear how a potential user might use the system. If we restrict the attention to the target users, we have to anticipate his or her behaviour. This concerns the intentions of the user, the navigation through the system, the activities started by the user, as well as the requested support. Users are not obliged to use the system. To the contrary, the system has to be designed in such a way that it becomes attractive for the target users. Only users who get the impression that the web information system is personalised to them, will make full use of the offered functionality.

The question why the system will be used is still closely related to the aspect of interaction. In fact, anticipating the users' behaviour includes reasoning about their intentions.

The questions where and when the web information system will be used now receive an important, yet still simple answer. Due to the universality of the web the system must be available at any time and at any place. This will have major implications on performance requirements.

We will further elaborate on this in Section 1.2. Our approach to the modelling of web information systems was first sketched in [222, 221, 720], then further developed emphasising the usage aspect in [711, 739, 529], content and functionality in [747, 526], pragmatics in [730, 732, 740], querying in [707], adaptivity in [708], collaboration in [91, 852], personalisation [722, 725, 737], reasoning [743, 709, 728, 738], quality assurance [93, 751], presentation design [583, 584], page generation [414], and various applications in e-business [398, 713, 710], e-learning [721, 90, 394, 744] and edutainment [736], and taken further towards data-intensive web services in [530, 531, 536] and service interoperability in [748, 749]. The strategic aspects of our modelling approach have been described in [586], the conceptual model in [716, 535, 726, 727], the pragmatics aspects in [735], the formal approach to adaptivity in [714, 734, 745], the applications to e-business and e-learning in [397, 557] and [742], respectively, and the extension towards data-intensive services, service interoperability and cloud computing in [532, 534].

1.1.2 Context Modelling

According to the “Langenscheidt-Longman Dictionary of Contemporary English” the *context of something* is “the situation, events, or information that are related ... [to this particular something] ... that help you to understand it better”. Therefore, we may distinguish different levels of contexts for web information systems. The distinction is based on what constitutes the situation, events, or information that define the context.

- The *application context* deals with the outside world and reflects the understanding of users and technical environment at the time the web information system is designed.
- The *WIS context* deals with the purpose of the system, i.e., which needs does it address, who are the target users, who are the stakeholders, etc. The WIS context deals with the information that is needed for a user to become interested in accessing the system.
- The *user context* deals with the information that is needed for a user who is using the system. The problem is to understand what a customer did while accessing the WIS. This includes his or her particular goals and reasons for not achieving them.

Let us discuss a bit more the issue of the user context focusing on how to support a user navigating through a WIS in order to solve a particular problem. This requires a user model being represented in the WIS, and used for determining appropriate help, while the user is operating the WIS. This view of context was emphasised in [399, 401, 89, 711, 527].

The concept of *context space* is a key conceptual tool for context modelling. In a large WIS a user may lose track of which information was already received and how to continue towards the intended goal. This implies that in a situation where the user may lose track the actual WIS state has to be mapped onto a point in the context space, which then is used to determine the best suitable means that helps the customer overcome the losing-track situation. Ideally, the point in the context space would contain condensed information about the path of the user through the system.

This path information depends on the type of the user. Therefore, in order to define an adequate context space we need a model of the target users. The association of a user type to a user has to be understood as a dynamic association, as some user characteristics might change throughout a session.

1.1.3 Large, Distributed and Cooperative Systems

Similar to enterprise information systems, a web information system should be considered as usually being very large in terms of the amount of data held in the system, the number and diversity of users, and the different navigation paths through the system. It should be expected that web information systems are even larger than traditional systems.

This size assumption implies that requirements for data-intensive information systems have to be taken into account. This includes the following:

- The whole data content of the WIS is structured, which implies using databases for this purpose.
- However, what is to be made available and visible for a particular activity by a user, has to be extracted from the database, which implies using views.

- The maintenance of data must be at the same standard as for traditional information systems, i.e., the functionality of a DBMS must be preserved within a WIS.

In addition, the size assumption has implications on the support for the users. This includes the following:

- As WISs are open they can be used by any user. However, these users should get the impression that the system adapts to their specific needs, i.e., the system has to be personalised for them.
- In addition, users should have the possibility to tune the system and adapt it to their specific needs. This implies that presentations should preferably be derived from content specifications and user-defined parameters.
- The functionality of the WIS must be task-oriented including the possibility to run several tasks at the same time, to interrupt or to split tasks, and to trace task progress.

Furthermore, while enterprise information systems may be distributed, a WIS usually is distributed. While in most of the cases this only refers to the universal access to the system via the web, it may also include the distribution of the WIS content itself. As a consequence of the size assumption it is quite natural to take also into account that the data is distributed. This implies the full support of distributed databases including distributed transactions and access separation, whenever this is needed.

Finally, some web-based systems are intended to be used by a group to achieve a joint goal. These cooperative systems add further difficulties with respect to communication, synchronisation, the support of “session”, and the use of the WIS as some kind of blackboard.

1.1.4 Categories of Web Information Systems

In the literature, several specific categories of web information systems have been introduced. The most important ones are e-commerce systems, e-learning systems, and information services. Unfortunately, there is a tendency to treat these categories no longer as specific application areas of web information systems, but to claim these to be independent subjects of their own. However, all these categories can be described by a pattern of the form $\mathcal{P}^W\mathcal{U}^A$ with the meaning that a provider \mathcal{P} provides something (W hat) to a user \mathcal{U} in order to support some activity A [586].

In the following chapters we will emphasize mainly the aspects that are common to all WIS. From time to time, however, we will refer to some of these categories as examples, mainly the following ones.

Electronic Business and Electronic Commerce

In this case the provider is usually a business, and the user is either a consumer or a business, too. What is provided are goods and services, and the main

activity is to purchase these goods. This leads to a pattern B^g2C^p or B^g2B^p , which are usually shortened to B2C and B2B, respectively.

Communities and Groups

In cooperative WISs the providers and users are members of the same group. The provided goods depend on the activity. For instance, in a conference system, the exchanged goods would be reviews, and the activity would be to discuss these reviews, which would lead to a pattern $\text{Group}^{\text{review}}2\text{Group}^{\text{discuss}}$.

Entertainment

In entertainment WISs we have a (commercial) game provider and players as users. The activity is playing, and the provided goods are games. This gives a pattern $\text{Provider}^{\text{game}}2\text{Player}^{\text{play}}$.

Identity and Personal Presentation

Identity WISs provide information for those who are interested in the company, club, individual, etc. This leads to a pattern

$$\text{Presenter}^{\text{information}}2\text{Interested-Party}^{\text{inform}}.$$

Learning and Edutainment

Learning WISs lead to a teacher to student communication pattern with knowledge provided and learning as the major activity [742]. This leads to a pattern $\text{Teacher}^{\text{knowledge}}2\text{Student}^{\text{learn}}$.

Information Services

Information services are similar to identity WISs, as they mainly provide information. However, the target group is usually larger. For instance, in a tourist information service the provider might be a city, and the major activity might be booking, i.e., we obtain a pattern $\text{City}^{\text{information}}2\text{Tourist}^{\text{book}}$.

It should be noted that most WISs, especially large size WISs, may have more than one provider, more than one category of user, and support a diversity of actions. For instance, a more general information service pattern would be

$$(\text{City}, \text{Hotel})^{\text{information}}2(\text{Tourist}, \text{Investor})^{\text{inform, book, invest}}.$$

More generally, the main activities to be done with a WIS can be described by verbs, and the pattern is a coarse indication of the *word field* of the verb. This links to a linguistic analysis in the development methodology. We will come back to this more general aspect in Part II when dealing with storyboarding.

1.2 Characteristics of Web Information Systems

A data-intensive information system that is realised in a way that users can access it via web browsers is called a *web information system*. On the surface, such systems are realised by a *web site*, i.e., a collection of web-pages, each of which is just a file that can be interpreted by the browser (e.g., Firefox, Internet Explorer, Safari, Chrome, etc.).

To a large extent such systems can be designed and developed along the same principles that apply to “normal” information systems. However, information systems that are used within enterprises are usually closed systems in the sense that they serve a particular internal purpose, and the users of the system are well known. The purpose and usage of the systems as well as the users are known in advance.

However, a web-based system is “open” in the sense that everyone who has access to the Internet and is able to use a web-browser can appear as a system user. Therefore, we have to pay more attention to a “mission statement” for the system, i.e., the important questions “Who will use the system?”, “Which user intentions and behaviour shall be supported?”, “Which technical devices will be used by the users?”, etc. have to be taken into account.

We will now look at these questions in more detail focussing on six different aspects: intention, usage, content, functionality, context, and presentation.

1.2.1 Intention

The intention aspect is a very general one centered around a mission statement for the system [586]. The primary question is: what is the purpose of the system?

For instance, in e-commerce systems the answer may be that the major purpose of the system is to sell certain products. In e-learning systems the purpose may be to provide learning material to students including useful hints and links to supplementary literature. In so-called “information services” the primary purpose may be to provide information about a city, a museum, a country, etc. to a potential visitor. It may also link to virtual tours. In group information systems, say a web-based system for a sports club or a scientific community, the primary purpose may be to provide information about special events to members or to use the system to attract new members.

The second question is this: Is there just one major intention or are there several minor intentions as well. For instance, in an e-commerce system a minor intention may be to promote a certain brand or to bind customers to the selling company. In commercial e-learning systems a minor intention may be to attract students to book further courses – in this case the system would be a mixture of an e-learning and an e-commerce system. In an information service system a minor intention may be to attract visitors and direct them to a booking system – again the system may be combined with an e-commerce system.

A third question associated with intentions concerns their time-scale. Some of the intentions may be long-term, others short-term. For instance, the minor intention in an e-commerce system to bind customers to the vendor may be a long-term goal, but the selling of particular products is short term. Similarly, a short-term goal in information services may be to advertise the city, region, institution, etc., whereas a long-term goal is to raise bookings by tourists.

The fourth question is a linguistic one and concerns the existence of metaphors that may describe the mission. For instance, as a major activity associated with e-commerce systems is “shopping”, many e-commerce systems provide a shopping cart as a visual metaphor, even if the products are not sold in shops using shopping carts. In an information service system for a museum the analog of a guide or a floorplan may be useful metaphors.

Electronic Business and Electronic Commerce

According to the definition of *commerce* the major purpose of an e-commerce system is to advertise and sell products of any kind. This implies that the system must support all sales, shipping and payment activities. The purpose of e-business is wider. It includes all sorts of businesses such as banking, joint ventures, exchange relations, etc. In general, the scope of e-business exceeds the scope of web-based systems.

According to this sales purpose the major goal of the system is to manage the sales business in a reliable way. Despite some myths about e-commerce, its use will hardly increase the overall volume of sales, but it does offer the advantage to the company of exploring new business channels. Being the first or the best may be an extra advantage for some time; being too late may lead to serious disadvantages.

A secondary minor goal of e-commerce is to bind customers to the company. If the electronic sales service turns out to be satisfactory to the customer, he or she may repeat visiting the system.

The area of e-commerce was probably the first to use icons that visualize metaphors associated with sales activities. Examples of such metaphors are the shopping cart, wish list, order form, cashier, etc.

Communities and Groups

Of course, the major intention of a community or group system is to serve as a communication means for members. So, there is a well-defined group of principal users who expect specific information from the system, communicate with other members via the system, handle their organisational affairs with the group, e.g., paying member fees, etc.

An additional minor intention may be to provide a window of opportunity for non-members to inform themselves about the group or community. In fact, this may lead non-members to become interested in joining the group or community.

Entertainment

The major content of entertainment WISs is to offer games, jokes, personal advice, etc. The major intention is to provide fun for the users. This is sometimes coupled with a commercial intention in the form of adverts or links to a group or e-commerce WIS.

Identity and Personal Presentation

Identity WISs are intended to present and advertise a company, private person, non-commercial association, club, etc. However, pure identity WISs are not very common. In many cases, the WISs link to e-commerce, entertainment, edutainment, etc.

Learning and Edutainment

E-learning is now nearly as popular as e-commerce and e-business. The providers range from universities over non-profit organisations to professional training institutions. Furthermore, sites of museums, exhibitions, etc. can be counted as learning WISs.

The major intention of a learning WIS should be to support learning. In some cases the slogan “life-long learning” is associated with a learning WIS. Thus, knowledge should be made available and self-control of the learners should be enabled. The spectrum ranges from systems that provide only knowledge sources to systems that are intended to simulate as much as possible the behaviour of a teacher. In some cases this major intention is coupled with a commercial goal.

In the case of so-called edutainment systems the provided knowledge is usually easy to grasp. The message is that learning can be fun.

Learning WISs benefit from several metaphors. For instance, for exhibitions or museums a floor plan or a guide may be useful metaphors. In e-learning classes the blackboard used for discussions can be used.

Information Services

Information services are probably the largest class of web information systems. These are used to provide information of all kinds. Examples are regional or city information services providing information about the region or city, respectively, including information about accommodation, food, latest news, special cultural services, sport information, etc.

The major intention behind information services is the information provision. Of course, a minor intention is advertisement in order to attract tourists, other visitors, investors, etc. Metaphors to be used in information services are news, dictionary, yellow pages, guides, etc.

1.2.2 Usage

Once some clarity with respect to the intentions of the web-based system has been obtained, the question arises by whom and how the system will be used [727, 726]. As web-based systems are open systems, it is important to anticipate the behaviour of the users. Therefore, it is necessary to first obtain an idea of the expected users. Of course, not all these users may be of interest to the system provider, but it is important to identify those users that are to be supported by the system.

For instance, a bank may want to offer a web-based system for financial investments. In this case the bank will be interested in users who intend to invest and to provide the necessary information for them. Any other user of the system can be neglected.

As it cannot be expected that all users act in the same way, the problem of classifying the expected users arises. This may lead to certain user profiles. Such user profiles may be determined by the different goals of the users, their different intentions, their different behaviour, their information needs, their levels of required support, etc.

So, the activity of *user profiling* will lead to a list of profiles of expected users who are to be supported by the system. This influences the content of the site's pages, their logical organisation, the enabled navigation links between these pages, and maybe even their presentation. More abstractly speaking, for each user profile we have to anticipate how the users will navigate through the system. If we denote a possible sequence of pages by the term *story*, then the most challenging problem is to determine these stories and to describe them in an abstract and integrated way. We refer to this problem as *storyboarding*. The pragmatics of storyboarding suggests a further refinement of the stories by *mini-stories*, which typically captures a small, self-contained, tightly connected set of scenes similar to a clip for movies.

For instance, continuing the example of an investment system, we may have users who do not want to take a big risk, while others behave almost like gamblers. The stories associated with these two profiles will be quite different, as users with the former profile should be directed to investment opportunities with more or less fixed interest rates, whereas users with the latter profile are likely to invest in stocks and shares.

Electronic Business and Electronic Commerce

Apart from e-commerce and e-business systems that are devoted to communication between businesses (so-called Business-to-Business or B2B systems) the users are normally unknown. In fact, the users can be any customers, and we talk of Business-to-Consumers (or B2C) systems.

With respect to user profiles all sorts of users are possible. Users may be well-informed regarding the offered products or may be just curious. Users may search for well-determined products or just explore what is offered. However,

we may assume that users share an interest in quality of the system. Quality criteria for users can be reliability, usability comfort, prices, security, privacy, etc.

Describing the stories can be based on the assumption that in most cases users just enter the site, search for a product or navigate through the products on offer and then decide to buy. So, the shop metaphor is quite useful.

Communities and Groups

In contrast to many other classes of web-based systems, community and group support systems are mainly used by a well-defined group of users. The user profiles within the group should be known. In most cases it can be assumed that no difference between user profiles of group members will be made, as information on how the system should be used can easily be made available to the members.

The service available to non-members can be reduced to navigation through some general information and most importantly accessing the information how to become a member.

Entertainment

The usage aspect is of minor importance for these sites. Users either take the offer as it is or leave it. No specific user support is provided.

Identity and Personal Presentation

As for entertainment sites there is no limitation on the expected users. However, as the site is meant to leave a positive impression on its visitors, the usage aspect is of much greater importance than for entertainment sites. The activities of storyboarding and user profiling are necessary to avoid users getting lost on the site, i.e., they do not find the required information within a huge collection of information, most of which they do not need.

Learning and Edutainment

The usage of a learning system depends on whether the control of the learning process is left to the user or the system. In both cases, however, it is assumed that the users are willing to learn and match the required prerequisites.

Users normally enter the site more than once continuing a specific learning programme. This requires some authentication mechanism, especially if the learning progress is controlled by the system.

Quality criteria are set by the teaching quality, and in the end, by the increase of knowledge on the side of the users.

Information Services

In information services it is quite common to expect all kinds of users. The assumption is that the users are normally unknown, however, users may also be familiar with much of the information provided. Thus, as in e-commerce systems, users may be just curious or well-informed, looking to satisfy well-determined information needs. Thus, all kinds of user profiles are possible.

The standard scenario is that users enter the site, search for or navigate to some information and leave again. If the system includes the possibility to book accommodation or tickets, then the user may also do such bookings.

Quality criteria for an information service are the availability of desired information and the ease of access for the user.

1.2.3 Content

The content aspect is central to the development of the system, as it concerns the question: “Which information should be provided?” As with most “normal” information systems this is coupled with the problem of designing an adequate database. However, the organisation of data that is presented to the user via a web-site is significantly different from the organisation of data in a database. So, organising the data content of the site means to investigate the decomposition, structuring and classification of data in such a way that the stories identified in the storyboard can be adequately supported.

For instance, in an information service the information about the attractions of a city, the possible accommodations, the actual events, etc. may be used to set up a database design following well-known principles. For the site it may appear to be useful to combine data about an event, e.g., a conference, with special accommodation offers linked to the event, and information about attractions that are part of the social programme coupled with the event.

Thus, modelling the content of a site has to be addressed on at least two levels: a logical level leading to databases, and a conceptual level leading to the content of pages. Both levels have to be linked together. Furthermore, in both cases abstraction mechanisms should be used. While such abstraction mechanisms are established in the area of databases, they are still a matter of research for web-based systems. The same applies to quality criteria.

Modelling site content must take into account that information must be presented in different ways to different users. This depends on the profile of the user, the communication channel, and the available devices. Modelling site content has to provide mechanisms to tailor the content automatically according to these parameters.

Electronic Business and Electronic Commerce

The content of an e-commerce or e-business site must be well ordered. The major component is some kind of product catalogue containing product descriptions, shipping mode descriptions, prices, etc. It may be useful to adopt

the metaphor of “catalogue” directly. Alternatively, a clear categorisation and classification of products is needed. This can be realised by adding an ordered table of contents or a search index to the content.

Besides the information about the products themselves, lots of additional information may be useful or required by the user. This information comprises sales conditions, special offers, self information, bestsellers, sales statistics, etc.

Communities and Groups

The major part of the content is made up by documents available to members or interested non-members. If the site is used for intra-group communication, then access to discussion fora, chat rooms, etc. can be part of the site’s content. Some forms for authentication, managing organisational affairs, and membership renewal or application, respectively, may be added as well.

Identity and Personal Presentation

The content of an identity can be almost everything. For company presentation sites the content may comprise history, philosophy, product spectrum, offers, etc.

Learning and Edutainment

The content of a learning site depends on the area that is to be taught. Learning sessions are used as structural means, and navigation through these sessions may be organised in linear form or as a directed acyclic graph.

Systems may be completely passive allowing only material to be read or downloaded. Other systems may involve upload mechanisms for assessment of the learning progress. According to the progress made, a system may even provide feedback to its user.

Information Services

The major content of an information service site is made up from up-to-date mass data, which is usually coupled with a lot of contextual information. In cases, where the services include booking systems, a categorisation of offers and self descriptions of the providers will be required as well.

1.2.4 Functionality

The functionality aspect is coupled with the question, whether the site should be passive or active. A passive site would only allow a user to navigate through the pages without any activity. For instance, a pure information service, e.g., a newspaper service, or an edutainment site, e.g., the site of a popular science

journal, may be passive. In these cases the major problem associated with functionality is to set up an adequate navigation structure.

In an active site, however, information would also be required from the user. From a conceptual point of view, the main purpose of functionality modelling is to identify functions that are available at a site to support the activities of the users, which were identified in the storyboard. Such functions can be system-specific functions in order to process user input or general support functions for searching, printing, marking, extraction, etc. For instance, a site may provide just an internal search function to search within the site, provide an interface to write ad-hoc queries to an underlying database, or simply embed a link to a standard external search engine.

Sites can be more or less passive, but nevertheless provide some functionality besides navigation links. For instance, the provision of a search facility or the use of a simple login form may not be counted as turning the site into an active site.

Electronic Business and Electronic Commerce

The major functionality offered by an e-commerce site is given by navigation and search, the possibility to collect and reposition products, and to check out at a cashier. Of course, the “cashier” is again a metaphor. In general, navigation and search lead to product descriptions, thus suggesting a decision to buy the product(s). The collection and repositioning allows several items to be purchased simultaneously.

Additional functionality can be provided by linking to co-operating e-commerce sites, supporting personal inquiries, e.g., via e-mail, supporting feedback, e.g., via wish-lists, etc.

Communities and Groups

Community and group sites support passive navigation and active cooperation, e.g., if the site is used for organising a conference organization. As there may be the group of authorised members and the group of interested non-members, authentication mechanisms are required to restrict the access for non-members. This leads to a discussion of security aspects for the site.

Entertainment

Also the functionality is limited. In the case of games, mainly internal navigation as inherent part of the game is provided. This may be coupled with help support, possible opponents, hints, rules, etc. As no real search functionality is needed, only an overview is given.

Identity and Personal Presentation

As only passive usage of the site is expected, the functionality is usually limited to a simple navigation structure. In particular, no real search functionality is needed. Many identity sites exploit a guestbook metaphor and closer contact via e-mail.

Learning and Edutainment

The functionality of learning sites mainly supports the navigation through the site, i.e., the navigation through the learning material. In contrast to other systems, this navigation is a long-term progress with usually many interruptions. More sophisticated systems provide system-driven repetition and feedback. Apparently, the functionality of such systems is still a matter of research. Also, personal information needs can be supported by providing an interface to e-mail.

In learning and edutainment systems that model museums, exhibitions, etc. the navigation through the system should be organised in a way that it resembles the walk-through in a “real” museum.

Information Services

Supporting passive navigation, which allows information to be collected, is the major functionality of an information service. This requires an overview of the complex navigation structure, which may be close to peer-to-peer navigation, should be given.

In addition, information services must provide reasonable internal search mechanisms, support personal inquiries via e-mail and collect feedback from the users.

1.2.5 Context

The context aspect deals with the context of the WIS with respect to society, time, expected users, and the paths of these users through the system. In Section 1.1 we already discussed a classification of contexts. According to this classification the contexts of the application and the system are already covered by the ‘intention’ aspect. So we focus on the usage context, which means that the usage of the system by a user defines the context for the user in a particular situation. The key question is how the navigation of the user can be supported to prevent him or her from losing track of his or her goals.

This question is connected to user profiling and storyboarding on one side and the data content on the other. That is, given a path of a user in the storyboard with a particular profile for this user, the question is how much of this information will be needed as context information, and how will this contextual information be made available.

The treatment of contextual information is largely independent from the category of the WIS.

1.2.6 Presentation

Finally, the presentation aspect concerns the final realisation by web pages. This depends on the support of technical end-devices such as computer screens, television, cell phones, etc. and set layout preferences.

For instance, when you look at various e-commerce sites, you will discover that some of the information is placed on the entry page. In particular, this concerns the access to the catalogue, the search facility, the currently offered specials and the access to the self information. This is partly done in accordance to results from marketing psychology trying to attract the customer to products he or she is not primarily interested in, and partly, because this gives the user the impression of a well-ordered site.

Throughout the site specific icons representing the shop metaphors are used.

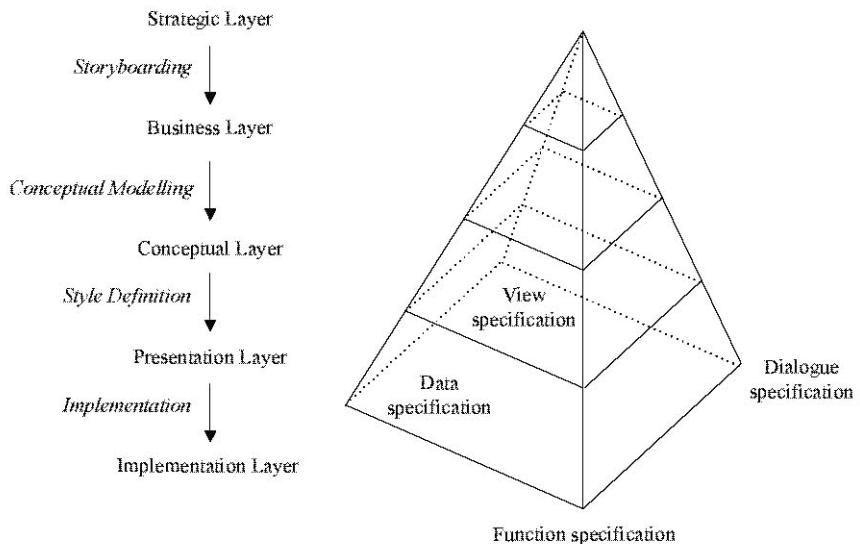


Fig. 1.1. Abstraction layers in Web Information Systems

1.3 Abstraction Layers in WIS Modelling and Design

The methodology for the development of web information systems is based on an *Abstraction Layer Model* [727], which is illustrated in [Figure 1.1](#). The general ideas of this model are as follows:

1. We identify several layers of abstraction. The top layer is called the *strategic layer*. It is used to describe the system in a general way: What are the intentions? Who are the expected users?

The next lower layer is called the *business layer*, which is used to concretise the ideas gathered on the strategic layer. This means to get a clearer picture of the different kinds of users and their profiles. This may also include the different roles of users and tasks associated with these roles. The major part of this layer, however, deals with the description of the storyboard. Stories identify possible paths through the system and the information that is requested to enable such paths. So the general purpose of the business layer is to anticipate the behaviour of the system's users in order to set up the system in a way that supports the users as much as possible.

The central layer is the *conceptual layer*. Whilst the business layer did not pay much attention to technical issues, they come into play on the conceptual layer. The various scenes appearing in the storyboard have to be analysed and integrated, so that each scene can be supported by a unit combining some site content with some functionality. This will lead to designing abstract web interaction types. The information content of the web interaction types must be combined to design the structure of an underlying database.

The next lower layer is the *presentation layer* which is devoted to the problem of associating presentation options to the web interaction types. This can be seen as a step towards implementing the system.

Finally, the lowest layer is the *implementation layer*. All the aspects of the physical implementation have to be addressed on this layer. This includes setting up the logical and physical database schemata, the page layout, the realisation of functionality using scripting languages, etc. As far as possible, components on the implementation layer, especially web-pages, should be generated from the description on the higher layers.

2. On each layer except the strategic layer, we identify two dimensions for the description of the system: *focus* and *modus*. The focus dimension distinguishes between local and global components; the modus dimension distinguishes between static and dynamic components. As this leads to four combinations, we distinguish between the following components:

- global and static components, which are addressed by a *data specification*,
- global and dynamic components, which are addressed by a *function specification*,

- local and static components, which are addressed by a *view specification*, and
 - local and dynamic components, which are addressed by a *dialogue specification*.
3. Each layer is associated with layer specific modelling tasks. The transition from the strategic to the business layer is associated with the activities of storyboarding and user profiling. The transition from the business layer to the conceptual layer is associated with conceptual modelling, which addresses database modelling, operations modelling, view modelling, and web interaction type modelling. The transition to the presentation layer is associated with the definition of presentation styles. Finally, the transition to the implementation layer is associated with all implementation tasks. The fact that these layers exist in the model and that the methodology is based on transitions between these layers does not imply that a development project must first finish the work associated with one layer before it can proceed to the next lower one.

In the following subsections we will discuss further several details of the abstraction layers.

1.3.1 Strategic Layer

As already stated the purpose of the strategic layer is to define the intentions of the web-based system and to identify the target users [586]. Intentions should be grouped into major and minor intentions. For each identified intention a time-scale should be determined as well.

For instance, a group system for organising a conference intends to provide

- general information about the conference, its history, its scope, its location and dates, etc.,
- specific information for authors how to submit papers, uploading and downloading facilities, the reviewing process, etc., and
- easy mechanisms for the programme committee to upload reviews, discuss about acceptance, generate feedback to authors, etc.

The identification of target users does not yet aim at classifying them according to their profiles, i.e., the questions *how* users access the system will not yet be addressed. However, distinguishing between internal and external users, users with specific access rights and obligations, thus playing a specific *role*, and identifying the *tasks* assigned to such users is considered as an activity on the strategic layer.

For instance, a group system for organising a conference may identify different user roles:

- The normal users are potential participants and authors of papers. They should access the general conference information and have an interface for uploading abstracts and papers.

- The programme committee members and especially the co-chairs need a specific access to the reviewing system as a subsystem, which allows papers to be downloaded, reviews to be uploaded and discussed, etc.
- The organisation committee members need a specific access to upload specific information of organisational nature such as registration and travel information.
- The administrator is the one who can grant and revoke access rights for the other users.

A common technique applied to these tasks is to have a brainstorming session. On this level, intentions, target users, roles and tasks should be described by informal documents using just text or tabular representations.

1.3.2 Business Layer

Based on the findings of the strategic layer the development process will proceed with the tasks of *storyboarding* and *user profiling* [727, 726, 735]. The emphasis is on anticipating the behaviour of the target users, i.e., how they will navigate through the system, which information they will need on their path(s) through the system, which information they will produce and enter into the system, and which actions they will carry out. Therefore, the activities on the business layer are the following:

- Analyse *roles* and *tasks* of the intended users;
- Classify users according to *user profiles*;
- For the different roles and profiles describe *stories*, i.e., navigation paths through the system;
- Integrate stories into *scenarios*, i.e., networks of scenes and transitions between them;
- For each scene describe the information the users consume and produce, i.e., information taken from the system and entered into the system, respectively;
- Describe the *actions* of the users for the different roles, tasks and profiles within the identified scenarios;
- Identify *metaphors* that may help a user to achieve familiarity with the system.

The user roles that were identified on the strategic layer already provide a first classification of users. This classification refers to the rights and the obligations of the users. For instance, certain parts of the system may be reserved to specific roles. In the conference organisation example that we used in the previous subsection, the paper reviewing subsystem is exclusive to the members of the programme committee (and the administrator), whereas the general information about the conference is open to all users.

Certain tasks are associated with each role. Identifying these tasks already gives an indication of different stories, as different roles should lead to

different navigation paths. It is also possible that an individual user can have more than one role. For instance, in the conference organisation example, each programme committee member is of course also a “normal user”.

Defining roles and tasks is again a brainstorming activity, which will result in non-formalised documents. These documents mainly identify the name of the role, a short textual description and a list of associated tasks. Each task should also be given a short description, but it must be described in more detail further on. This will lead to stories.

For each role the users have to be further classified leading to user profiles. User classification can be done by identifying several dimensions that are useful for the description of the users’ behaviour and using scales for each of these dimensions. Each possible combination of values taken from these scales could define a user profile, but it may be advisable to combine some of them. Thus, user profiles could be described by a matrix-document.

For instance, in the conference organisation example the users could be classified according to their interest in different parts of the conference including the social programme. In an information service system dimensions for user profiling could be the familiarity with technical systems, the educational background, the preferences with respect to information density, etc.

The roles and user profiles tell us who will use the system. Story boarding will analyse how the different users will navigate through the system. In practice, we may think of a story as a sequence of web-pages visited by a user. However, storyboarding does not start on the level of realised pages. It abstracts not only from presentational issues, but also to a large extent from technical issues. The business layer is still dealing with setting up the system in a way that non-technical stakeholders will be able to assess whether their goals can be met.

Thus, instead of talking about pages we use the abstract term of a scene. Story boarding identifies useful scenes and transitions between them. This can be described by graphs or transition matrices. Such graphs can be easily sketched in brainstorming sessions and interviews with stakeholders. Graphs, however, do not exhaust storyboarding. For each scene we may ask

- which information has to be provided to the user (this will be called *information consumption*),
- which information is to be requested from the user (this will be called *information production*),
- which *actions* can be carried out by the user to navigate to a successor scene including the possibility that the successor scene is the same scene again, and
- which *metaphors* are useful to support the actions associated with a scene.

Providing information consumption, information production and actions leads to enriched graphs or matrix representations. Furthermore, each transition can be coupled with information that is communicated between the two scenes. For the target scene this is *context information*.

The most difficult activity in storyboarding concerns the integration of individual stories. Scenes that have nearly the same information consumption but appear in different stories could be combined into one scene. This leads to further enriched graphs (or matrices), which are called *scenarios*.

Finally, scenarios can be described with various levels of detail. Usually, we start with (non-integrated) scenarios, in which a lot of information is left out. Details are added during a process of information gathering. This may lead to combining scenarios, introducing branches, replacing scenes by complete sub-scenarios, etc. These activities are subsumed under the notion of *story tuning*.

Story tuning has achieved its objectives when all the scenes in the scenarios can be transformed into *web interaction types*, which will be discussed in the context of the conceptual layer (see the next subsection for further explanation). In fact, the web interaction types are sufficient to describe the system. However, in order to determine them storyboarding and tuning are indispensable.

How is storyboarding related to the dimensions that we discussed in the previous subsection? Locally, we have to consider the scenes appearing in the scenarios. The static part (view specification) of the scene is determined by the consumed and produced information, whilst the dynamic part (dialogue specification) is determined by the actions associated with a scene. Globally, we consider complete scenarios. The dynamic part (function specification) is determined by the stories, whereas the static part (data specification) should be a description of all the information used in a scenario, which is left implicit on this layer.

The results of these activities make up the *Storyboard*, i.e., the description of the business layer. Details on how to describe the storyboard will be handled in Part II (Usage Modelling).

1.3.3 Conceptual Layer

Based on the scenarios that are defined on the business layer the development process continues with a *conceptual modelling* activity aiming at the definition of web interaction types [727]. However, web interaction types can be defined already during story tuning. The emphasis of the conceptual layer is on analysing the data processed in the scenarios and the actions on these data in order to describe the content and functionality aspects in a formalised way, and to develop database support. Therefore, the activities on this layer are the following:

- Group suitable data and specify operations for the actions, i.e., specify *content* and *functionality*.
- Specifying content and functionality leads to defining *web interaction types*.
- Restructure the data in order to define connections to databases.

- As database design follows different objectives (no redundancy, optimized access), the content specification should lead to *views*, i.e., transformations which turn the content of a database into the content of a web interaction type.
- Extend web interaction types in a way that they can be tailored to different users, different end-devices and different communication channels without designing multiple sites.
- Link the web interaction types with the scenarios and ensure that all scenarios are adequately supported.

With respect to the static component, i.e., the data, the various scenes will be analysed and the information consumption will be formalised using data types. This gives a local view on the data that is present in the system. For example, a scene in our conference organisation example (associated with the role of a programme committee member) might be devoted to looking at a completed review. So the information consumption is just the description of reviews. The analysis may lead to describing the parts of the review such as paper title, category, subject area, confidential comments, positive aspects, negative aspects, assessment of quality, assessment of readability, etc. This can be described in an abstract way using data types such that all possible reviews would become instances of that data type. Later on, such a data type will be called a *content type*.

Globally, however, we may need to organise the data in a different way. For instance, in the database of our conference organisation system we might find additional information on the programme committee member and additional reviewer(s) of a paper that are not visible to all programme committee members. However, it must be possible to map the database to the content type. Such a mapping is called a *view*. In our example we would simply have to extract just the information on the review.

With respect to the dynamic component, we would have to add operations. Locally, these operations are determined by the user actions associated with a scene. Globally, we would need operations on the database — to be precise, these operations will be *transactions* — which realise the operations on the web interaction type. Of course, the information production, i.e., the data requested from the user, must become the input of the operations.

How is conceptual modelling related to the dimensions that we discussed in connection with the abstraction layer model? Globally, we now provide a database schema for the static part (data specification) and database transactions (function specification). Locally, we have the analog of web interaction types. The static part (view specification) is defined by views on the database schema; the dynamic part (dialogue specification) is given by the operations associated with a web interaction type.

The result of this layer will be a *web interaction schema*, i.e., a collection of web interaction types, which adequately represent the scenarios. Details on web interaction types will be handled in Part III (Conceptual WIS Modelling).

1.3.4 Presentation Layer

The conceptual layer has produced the collection of web interaction types. The content and functionality of a system is already fully determined by the web interaction types. It has also been verified that the intention and usage of the system is adequately supported by the web interaction types. So, the remaining problem is the presentation, which is addressed on the presentation layer [586].

In general, a web interaction type defines a set of *web interaction objects*. The actual web interaction objects depend on the state of the database. Each of these web interaction objects is an abstract representation of a web-page. So, the major activity on the presentation layer is to specify style options for the pages. This will be done by enhancing the web interaction types with the style options. Some details concerning the presentation layer will be handled in part III (Presentation Modelling).

1.3.5 Implementation Layer

The implementation layer deals with all activities that are left to transform the presentation enhanced media schema into an executable and accessible web-based system. As a result of the higher layers we obtain a web interaction schema, which consists of a database schema and a collection of web interaction types. Each web interaction type is defined by a view on the database schema and operations. Furthermore, the web interaction type is associated with presentation style options. Therefore, the activities on the implementation layer are the following:

- Define a logical database schema and implement it by a database management system.
- Define the views of the web interaction types using the database query language.
- Generate web pages (HTML-documents) from the web interaction types and the style options as far as possible.
- Implement the operations on web interaction types using scripting languages.

Part IV (Web Information Systems Development) will discuss some aspects of the implementation. We should point out that the methodology aims at generating web-pages out of web interaction types instead of writing them by hand, similarly to how it is quite well understood how to generate a logical database schema out of a conceptual one. The process of generating HTML-pages can be supported by the using of XML as an intermediate language. Similarly, style options could be translated into the eXtensible Stylesheet Language (XSL). Generating HTML pages out of XML- and XSL-specifications is already supported by tools.

1.4 Bibliographical Remarks

1.4.1 General Literature on Web Information Systems

The literature on website development is very rich. Thus, this part will be inheritably incomplete. Many of the conferences and journals have a section on web information systems and added knowledge to practical realisation of websites. It is however surprising that the literature on real life case studies is rather limited. Real large web information systems are either firmware of companies such as CoreMedia or have been developed within a consulting business or are company software supporting the business processes at a company.

We restrict our guide to literature to some relevant literature within the proposed approach. Monographies that cope with some additional aspects are, for instance, [144, 158, 252, 367, 390, 603, 683, 857, 893].

A good starting point to website development methods is the survey [766]. An early source to website development are the books by D. Siegel [784, 785, 910]. Web information systems attract a lot of research. However, many problems remain open [843, 823, 825]. Classical database systems are human-active and DBMS-passive systems (DPHA) in the classification of [138]. Monitoring applications are DBMS-active and human-passive systems (DAHP), see e.g., [490, 945]. They can be extended to DBMS-active and event-pre-prepared systems (DAEP) [410, 365, 859, 871, 941]. WIS systems are typically *DBMS-active* and *human-active systems* (DAHP) [409, 420]. A brief survey on our approach is given in [201].

1.4.2 The Co-Design Framework

Traditional software engineering and information systems engineering (e.g., [537, 678]) is structured into requirements analysis and definition, systems design, systems implementation and testing, and systems operation and maintenance. It may be extended to application domain description, the requirements prescriptions, and finally the systems specifications [97, 98, 314]. For web information systems the traditional approach suffers in three obstacles: late integration of architectural decisions, neglecting user expectations, and late implementations. Web information systems follow pre-defined three-tiered architectures. They also must consider expectations, profiles and portfolio of a large variety of users. Additionally, users expect an early involvement into the development and an early evaluation of the system.

Database system design and development has been concentrated for a long time around data structuring. This orientation was based on logical and physical independence for a database specification and on universal programming features. Since database management systems could only provide a limited support for integrity maintenance, normalisation as a specific kind of optimisation has been developed. Functional dependencies and multivalued dependencies could be used to vertically decompose relational structures into structures without these dependencies, however at the price of referential inclusion

constraints [46, 181, 386, 500, 542, 549, 637, 822, 823, 930]. Normalisation can either be based on vertical decomposition or horizontal decomposition [637, 822] or deductive normalisation [822, 823]. Normalisation often results in structures that are again composed to denormalised ones during physical design [67, 136, 235, 508]. The database practitioner had to realise that functionality matters and that structuring and functionality must be co-designed [107, 235, 778]. Otherwise, already integrity maintenance and enforcement becomes a nightmare.

The co-design approach¹ of integrated development of structuring and functionality (additionally also distribution based on protocol engineering [439]) [221, 726, 709, 739, 741, 824, 823, 830] extends modern software engineering approaches by explicit consideration of the application domain [856]. Furthermore, it emphasises the intention dimension, which can be specified on the basis of life cases [732, 735]. Classical co-design of structuring and functionality is discussed in [191, 192, 918]. Interactivity is also discussed in [69, 448]. WIS specification is oriented towards systems that are easy and intuitively to use. [727, 830] extend these approaches by (1) explicit consideration of user expectations, profiles and portfolio and (2) by storyboards and story spaces. Particular design is oriented on the needs of the user [331, 407, 630, 635]. The six W questions can be extended to the W*H framework [175].

The literature for development of database structuring is fairly rich, e.g., [61, 80, 549, 916]. A number of comparisons have been proposed, e.g., [300, 486]. Interactivity development is a novel area of research. A number of groups, e.g., [56, 144], proposed methodologies for website design. None of the other methodologies has ever been evaluated. The integrated development of structuring and functionality is novel. Distribution development is mainly based on implementation steps. Website development is mainly based on logical or physical specification.

The advance of web information systems resulted in a different treatment of co-design [726]. Functionality is provided on the basis of specific operations. Ad-hoc querying might be provided but is not the central issue. Functionality in websites might be concentrated around specific search techniques, import/export features, navigation and linking, security and privacy, database access, and database-backing content consideration. The derivation of appropriate interfaces for each web page became now the main obstacle. Interactivity thus requires sophisticated support especially for the case of mass access to a given data-intensive web page. A first solution is the introduction of sophisticated view techniques in such a way that each web page is supported by a specific input view and a specific data delivery view, e.g., [430, 607]. View data generation may become now a performance trap for data-intensive websites. Therefore, a second solution has been based on materialised views

¹ The notion of WIS co-design inherited research on hardware-software co-design of systems, e.g., [37, 139, 464, 680, 903].

similar to data warehousing solutions [475, 488, 494, 578, 682]. The optimal solution is however co-design of structuring, functionality, and interactivity.

The co-design framework has also been elaborated within the predesign approach [239, 556] and for the KCPM and NIBA projects [237, 238, 240]. The predesign layer is similar to the business user layer in co-design approaches. The presentation layer explicitly treats the presentation side of websites. It is based on the conceptual layer and on interaction objects.

The categorisation and the six dimensions of websites are discussed in detail in Sections 5 and 6. We thus provide references in these sections.

The abstraction layer model is an element of the co-design development methodology [821] that is discussed in detail in Section 11. A number of models have already been proposed for Computer Supported Cooperative Work (CSCW) systems such as coordination theory [545], activity theory [391], task management approaches [461], action/interaction theory [231], and object-oriented conceptual models [816]. Co-design of WIS must overcome the classical legacy program and can be extended by modernisation, migration, and evolution approaches [156, 172, 319, 369, 368, 426, 679, 709, 726]

Key Messages

The **Co-Design** of web information systems must address

- the *intention* of the system answering what the system is to be used for;
- the *usage* anticipating who will use the system in which way and for which goals;
- the *content* of the system specifying which information shall be provided or collected;
- the *functionality* of the system specifying which actions can be performed by users of the system;
- the *context* of the system capturing how users are to be supported in order to remain targeted on their goals;
- the *presentation* of the system addressing layout and playout supporting the needs and preferences of the users.

The design and development of web information systems must consider **abstraction layers**, and on each layer address local and global as well as static and dynamic aspects:

- The *strategic layer* has to specify the intentions and target users.
- The *business layer* has to deal with the system usage and cover storyboarding and user profiling.

- The *conceptual layer* has to deal with the conceptual modelling of content, functionality and context, for which web interaction types will be developed.
- The *presentation layer* has to specify style options for the layout, for which the screenography method will be developed.
- The *implementation layer* covers all activities left to transform the design into an executable web information system.

Part II

High Level WIS Design – Strategic Analysis and Usage Modelling with Storyboarding



Strategic WIS Modelling

The goal of this chapter is to discuss modelling on the strategic layer of a WIS, an issue that is often enough neglected. Strategic modelling comprises two parts. The first one addresses a very general characterisation of the system in terms of its content, functionality, context, usage and presentation. This will be investigated in Section 2.1. The second part addresses pragmatic guidelines for strategic modelling, which we handle in Section 2.2.

The strategic characterisation of a WIS starts from the very general question what the WIS is about, i.e., the purpose(s) of the system, and what are criteria for the WIS being successful. The general answer to these questions gives rise to an informal *mission statement*, and a characterisation of the *brand* of the WIS. The latter one will follow the general classification scheme for WISs that we introduced in Chapter 1.

Going more into details we first explore the kind of content that is to be presented in the WIS, and the kind of functionality with which this content can be accessed, customised to the needs of particular WIS users, and updated. This defines the *utilisation space* of the WIS.

We then explore the *utilisation portfolio* to gain even more details. This means to model the users (or actors) who will use the WIS, their goals, i.e., why they are supposed to use the system, and the tasks that have to be performed to reach these goals.

The fourth and last part of a strategic WIS model are general rules for the formation of the WIS presentation. These address the gestalt, atmosphere and progression of the system based on knowledge about the cognitive perception of form, colour and other style elements.

Taking these parts together we should keep in mind that the role of strategic modelling is to lay out the plan for the whole WIS without drifting into technical details. As a consequence, the techniques applied on this level will be rather informal, whereas formalisation will be achieved on lower levels of abstraction. The rationale behind this approach is an observation made in theoretical linguistics: whenever a complex construction has to be explained, humans first think in terms of concepts, which are then mapped to a linguistic

construct and only finally translated into sentences. Carrying this idea over to WIS development means to first lay out the fundamental concepts that are to be captured by the system, then map them onto a conceptual model, before finally approaching an implementation using common available technology.

In principle, the modelling of a WIS on the strategic layer must also address the technical choices for the implementation. However, these will only come into play much later, so we defer the discussion of these aspects to part IV.

2.1 General Characterisation of a WIS

The strategic layer of a WIS is meant to set the target for the models that are to be developed at the lower layers. We will characterise a WIS by

- a mission statement describing in general terms what the WIS is about,
- a utilisation space describing content, functionality and context,
- a utilisation portfolio describing actors, goals and tasks,
- the utilisation context, and
- general principles describing the ambience and desired atmosphere of the WIS.

The last part will guide the formation of the WIS. Issues of strategic WIS modelling have been discussed before in [711, 586]. In addition, the utilisation context was mentioned briefly in [527, 530].

2.1.1 Mission Statement and Brand

The *brand* of a WIS adds details to the classification scheme for WISs introduced in Chapter 1. This classification scheme has the form $\mathcal{P}^{\mathcal{W}}\mathcal{U}^{\mathcal{A}}$ and represents in an extremely terse form the following very general information:

- \mathcal{P} stands for “provider”, and thus indicates which role the system plays. Examples used in Chapter 1 were business, teacher, city, presenter, etc., which indicate very roughly what kind of content can be expected from the system. For instance, if the provider is a bank, the provided services will most likely center around accounts, investments, savings and loans.
- \mathcal{W} stands for “what”, and thus adds more detail to the kind of content offered by the WIS. Examples used in Chapter 1 were goods, games, information, knowledge, etc. For instance, if the provider is a bank, the provided content may just be accounts, investments, savings, loans, and mortgages.
- \mathcal{U} stands for “user”, and thus indicates to whom the services offered by the WIS are directed. Examples used in Chapter 1 were customer, student, tourist, investor, etc. For instance, if the provider is a bank, the users are probably just the customers, enterprises or other banks.

- \mathcal{A} stands for “actions”, and thus indicates the functionality of the WIS offered to its users. Examples used in Chapter 1 were purchase, dicuss, play, learn, inform, book, invest, etc. For our example of a bank offering accounts, investments, savings, loans, and mortgages possible actions can be apply_for_loan, apply_for_mortgage, set_up_account, buy_stock, etc.

The brand is usually the result of a brainstorming activity discussing the what, whom and for_whom of the WIS. The aim is to fill these general place-holders \mathcal{P} , \mathcal{W} , \mathcal{U} and \mathcal{A} with meaningful terms that describe the WIS in very general, terse terms. We shall look at this brainstorming activity in more detail in Section 2.2.

The brand gives a rough picture of the content and functionality of the WIS and its users using only descriptive keywords. This is, however, a valuable source of information for refinement using linguistic methods. We will look at these pragmatic methods for strategic analysis also in Section 2.2.

The *mission statement* complements the brand by an informal, textual description. Each of the actions in the brand are taken as the major tasks. For each of them the mission statement describes which types of users are involved, which activities they are supposed to execute, which content will be provided for them and requested from them, and what will be the results of these activities. However, no attempt is made to decompose the tasks or to refine them, as this is left to storyboarding (see Section 5).

Furthermore, the mission statement contains metaphors that turn out to be adequate for describing the activities associated with the WIS. These metaphors refer to the content and functionality keywords used in the brand.

In addition to its descriptive character the mission statement also has an explanatory character in the sense that it contains the reasons for setting up the WIS. That is, the mission statement will describe what the major and minor purpose of the system is, how each task will contribute to these purposes, and what the benefits of the system for the provider and the users will be.

Example 2.1. Let us consider the example of a WIS that deals with loan applications. In this case the provider is a bank, and the content will be centered around (personal) loans and mortgages. The only users we think of are customers, and the tasks they execute are applications for loans and mortgages, respectively. This leads to the following brand:

bank^{loan, mortgage}2customer^{apply_for_loan, apply_for_mortgage}

The mission statement is the following informal explanation for the brand:

The mission of the system is to provide on-line access for customers to personal loan and mortgage applications. The system will provide information about the available types of loans including conditions for

repayment, i.e., principal and interest, conditions for creditworthiness, and intended purposes of the loans. For mortgages this further includes securities and conditions for bailsmen. The information about loans will be complemented by easy loan examples. Then the system will allow customers to enter their personal data, select the appropriate loan, and check whether their personal finances are in accordance with the rules for repayment.

The major purpose of the system is to open an additional sales channel, which is addressing mainly those customers who are capable of dealing with electronic systems, do not need intensive advice, and therefore prefer the convenience of avoiding personal contact at a branch office. A secondary purpose is to improve the efficiency of banking in the loan sector.

The expected benefits of installing the system for the bank are a closer binding of customers to the bank, the possible attraction of new customers, and an improvement of cost efficiency, while at the same customers benefit from increased availability of bank services.

In general, it is sufficient to formulate the mission statement using free-form text as in Example 2.1, but it is also possible to use semi-formal structured text. In doing so the brand and mission statement take the following form:

Content:	\langle list of content items \rangle
Users:	\langle list of users \rangle
Tasks:	\langle list of tasks \rangle
Major Purpose:	\langle textual description \rangle
Minor Purpose:	\langle textual description \rangle
Benefits:	\langle textual description \rangle

Furthermore, we obtain the following informal description for each of the tasks:

Task:	\langle task name \rangle
Description:	\langle textual descriptor \rangle
Participants:	\langle list of users \rangle
Required Content:	\langle list of content items \rangle
Produced Content:	\langle list of content items \rangle
Result:	\langle textual descriptor \rangle

Example 2.2. Using the tabular semi-formal description, we can rewrite the brand and mission from Example 2.1 in the following way:

Content:	loan, mortgage
Users:	customer
Tasks:	apply_for_loan, apply_for_mortgage
Major Purpose:	open an additional sales channel address technology-experienced customers address informed, goal-oriented customers
Minor Purpose:	improve banking efficiency in loan sector
Benefits:	closer binding of customers attraction of new customers improvement of cost efficiency increased availability of bank services

Furthermore, we obtain the following informal descriptions for the task apply_for_loan:

Task:	apply_for_loan
Description:	The system will provide information about the available types of loans including conditions for repayment, i.e., principal and interest, conditions for creditworthiness, and intended purposes of the loans. The information about loans will be complemented by easy loan examples. Then the system will allow customers to enter their personal data, select the appropriate loan, and check whether their personal finances are in accordance with the rules for repayment.
Participants:	customer
Required Content:	list_of_loans, loan_conditions, loan_purpose
Produced Content:	loan_application, customer_data
Result:	confirmed_loan_application

The description of the task apply_for_mortgage will look similar.

2.1.2 Utilisation Space

The term “utilisation space” is used as a metaphor to characterise the WIS as a space, through which a human user can navigate. As such it has to cover mainly the content, functionality and context in general terms. The goal is to enable optimal orientation in the utilisation space, such that searching and finding information needed for certain tasks will be facilitated.

The type of content is already characterised by the brand, to be precise by its what-part. This gives a set of nouns describing the content in coarse terms. Similarly, the what_for-part of the brand gives verbs describing the functionality, i.e., what to do with the content.

The utilisation space will now add details to content and functionality, set the nouns and verbs used in the brand into relation, and place both into a utilisation context. This will be done in the following way:

- Refine the content keywords and place them in semantic relationships. These relationships can capture specialisation, part-of relationships, or associations of global context with details. They indicate navigation facilities and order principles among the content. Word fields are a valuable tool for the refinement (see Section 2.2).
- Refine the functionality keywords to discover various facets that can be placed in semantic relationships in the same way as the content. Again, word fields are a valuable tool for the refinement.
- Relate the functionality with the content, i.e., specify in which context a particular content is needed, i.e., which content is needed by which activity, which content is produced by which activity, in which order (if any) the content will be used by an activity. In doing so, we obtain a progression model for the functionality.

Semantic relationships for content – and similarly for functionality – can be represented by rooted trees, where the root is defined by a keyword taken from the brand. Thus, we can obtain the following more detailed description of content items:

Content Item:	$\langle \text{name} \rangle$
Derived From:	$\langle \text{content item} \rangle$
Relationship:	$\langle \text{description} \rangle$
Usage:	$\langle \text{list of tasks} \rangle$
Description:	$\langle \text{textual description} \rangle$

In the same way we obtain more detailed description of tasks:

Task:	$\langle \text{task name} \rangle$
Derived From:	$\langle \text{task name} \rangle$
Relationship:	$\langle \text{description} \rangle$
Description:	$\langle \text{textual description} \rangle$
Participants:	$\langle \text{list of users} \rangle$
Required Content:	$\langle \text{list of content items} \rangle$
Produced Content:	$\langle \text{list of content items} \rangle$
Result:	$\langle \text{textual description} \rangle$

The description of tasks may be extended by

Starting situation:	$\langle \text{list of situation parameters} \rangle$
Closing condition:	$\langle \text{acceptance constraints} \rangle$
Execution context:	$\langle \text{textual characterization} \rangle$

Typical situation parameters are preconditions and triggering conditions for task enactment, priorities among tasks, frequency of tasks, and repetition patterns. Acceptance constraints are used for the internal control, whether a task can be considered to be completed. The execution context is used for the description of duration restrictions, foreign interaction, scope, and robustness.

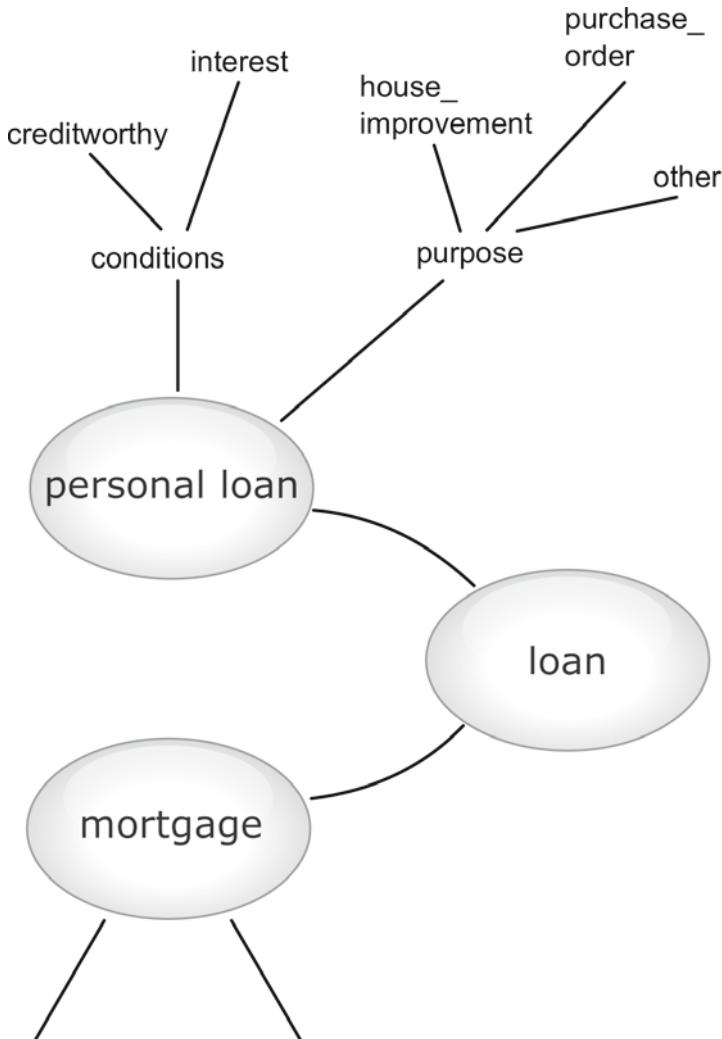


Fig. 2.1. Semantic tree of content items in loan application

Example 2.3. Figure 2.1 represents a tree of content items with root loan, which is specialised by personal_loan and mortgage. Details for mortgages have been omitted, but for personal loans the three decisive facets – conditions for obtaining the loan, i.e., creditworthiness, conditions for loan repayment, i.e., life span, principal and interest, and loan purpose have been indicated.

Similarly, Figure 2.2 represents a task tree with root apply_for_loan, which again is specialised by apply_for_personal_loan and apply_for_mortgage. For the latter one further details have been omitted. For apply_for_personal_loan

the components contributing to selecting terms and conditions, outlining personal finances, declaring the purpose of the loan, and entering customer details are shown in the tree.

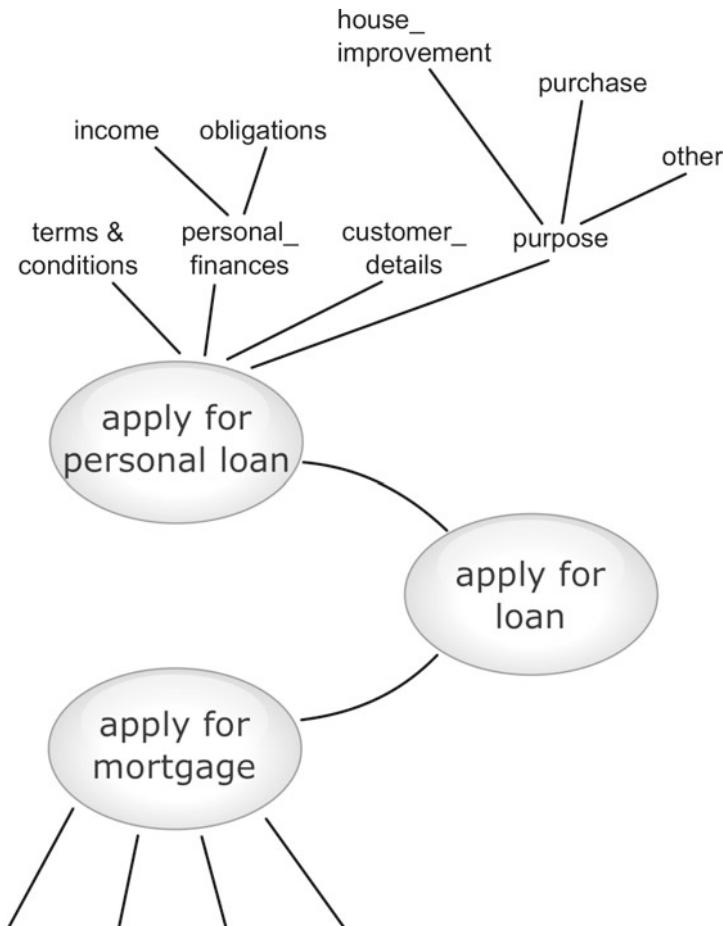


Fig. 2.2. Semantic tree of tasks in loan application

2.1.3 Utilisation Portfolio

The utilisation portfolio complements the utilisation space emphasising the whom-part of the brand. Thus, it is mainly concerned with the WIS users, for which we will later adopt the term “actor”, their goals and the tasks that have to be executed to achieve these goals.

Tasks correspond to the actions in the brand and their refinement in the utilisation space. So we can assume a task hierarchy emphasising specialisation between tasks and decomposition of tasks into subtasks, as long as these can be described in a simple way.

The users used in the brand and mission statement will be roughly classified according to roles they have with respect to the WIS. Each role has particular goals, and each of these goals corresponds to a task that is meant to achieve this goal. This does not mean that the task has to be executed by the user in this role; it may well refer to tasks executed by users in other roles. Tasks are broken down into subtasks to a level that elementary tasks can be associated with a single role. In addition, subtasks should refer to subgoals.

Furthermore, we obtain dependencies between goals, e.g., being a subgoal, a specialisation, or any other kind of dependency. Thus, we complement the informal description of the system by adding goals:

Goal:	$\langle \text{goal name} \rangle$
Derived From:	$\langle \text{goal name} \rangle$
Relationship:	$\langle \text{description} \rangle$
User:	$\langle \text{role name} \rangle$
Description:	$\langle \text{textual description} \rangle$

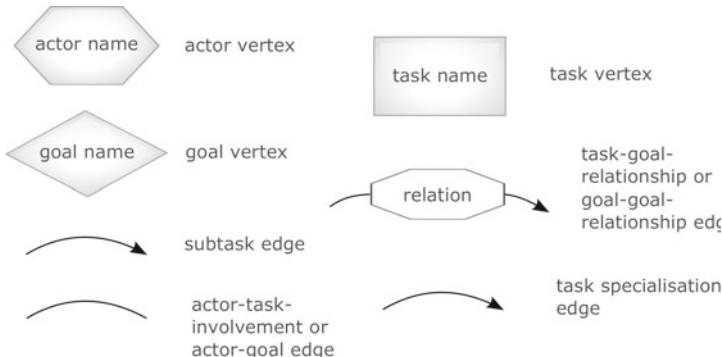


Fig. 2.3. Legend for task-goal graphs

The relationship between tasks, roles and goals can be represented in a graph, which we call a *task-goal graph*. In these graphs we have three different types of vertices for actors, tasks and goals, respectively. Furthermore, we have five different kinds of edges for task-goal relationships, involvement of an actor in a task, goal-goal-relationships, as well as for subtasks and task specialisation. [Figure 2.3](#) shows the legend for such graphs.

Example 2.4. The task-goal graph in [Figure 2.4](#) illustrates goals, tasks and actors in a loan application. In this case the goal buy_new_car depends on

obtaining a personal loan, i.e., on the goal `personal_loan`. Sufficient for achieving this goal is the successful execution of task `approve_personal_loan`, which has to be done by a bank clerk. However, this task will only be triggered by a task `apply_for_personal_loan` to be executed by the customer. This task is thus necessary for the goal. Furthermore, the task decomposes into four subtasks `select_conditions`, `enter_customer_details`, `declare_purpose` and `set_up_budget`.

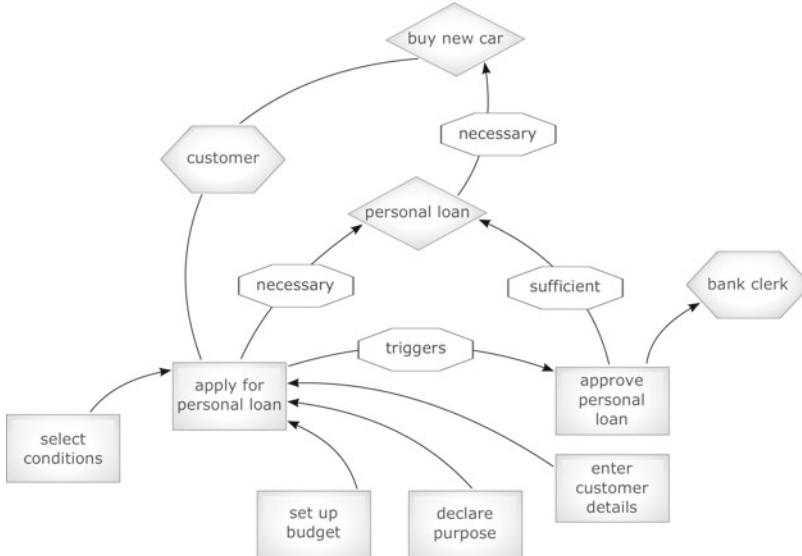


Fig. 2.4. Task-Goal graph for loan application

A valuable tool for setting up the utilisation portfolio is communication analysis, which addresses how a user will communicate with the WIS and why this communication is the best for the provider and the user. We shall discuss this in more detail in Section 2.2.

2.1.4 Utilisation Context

The decisive criterion for orientation in the utilisation space is the preservation of context. This can be achieved, if for each task a mental model can be constructed. This model captures the progression of content perception over a time axis. It represents a system map for the user showing information that has already been used and processed.

For strategic modelling it is important to reflect whether the stages that represent perception of content are logically connected. It is also important to see that later stages enable the possibility to be redirected to information that was already processed earlier.

If available, metaphors may help to set up this mental model. For instance, the “desktop” metaphor is a well-established tool that has significantly contributed to ease the use of computer technology by technical laypersons. Similarly, the “shop” metaphor is frequently used in commerce applications. We will discuss metaphors in Section 2.2.

Example 2.5. We may regard the task of personal loan application in the loan application system from Example 2.1 as a matching problem. We have to match the needs of a customer with the purposes of the available loans, and the payments arising from loan conditions with the payment latitude of the customer, which is determined by creditworthiness conditions set by the bank and the personal finances, i.e., income and obligations, of the customer.

Thus, a suitable mental model consists of the set of loan options, i.e., loan type, conditions and purpose, and the implications of each option with respect to the payments. This model develops over time in a way that non-suitable options are deleted first, then a selection is made, and finally organisational data are added to turn the selected loan option into a full loan application. This also indicates the required functionality, which must comprise selecting information about loan options, tentative and affirmative selection of such options and decision aids.

As already discussed in Chapter 1 each characteristic aspect of a WIS can be associated with a specific *context*. In general, we distinguish between the following different kinds of context:

- The *general context* associates certain characteristics of mission, utilisation space, utilisation portfolio, and general principles to other characteristics of the WIS.
- The *usage context* associates scenarios that describe the behaviour of users with other such scenarios.
- The *website context* specialises scenarios based on the supported environment, the provider and software and hardware environments.
- The *runtime context* permits the incorporation of the current situation and the information on the current usage of the WIS into the offered content and functionality.

The different kinds of context permit a layering and stepwise incorporation of context information into the WIS based on the abstraction layer model. The general context is specified by content items, functionality, actors, goals and tasks as shown in [Figure 2.5](#), thus can take the following form:

Content context:	\langle list of related content items \rangle
Functionality context:	\langle list of functionality \rangle
Actor context:	\langle list of cooperating actors \rangle
Goals context:	\langle list of related goals \rangle
Task context:	\langle list of related tasks \rangle

The other kinds of context will be discussed in Chapter 3 and Section 8.2.

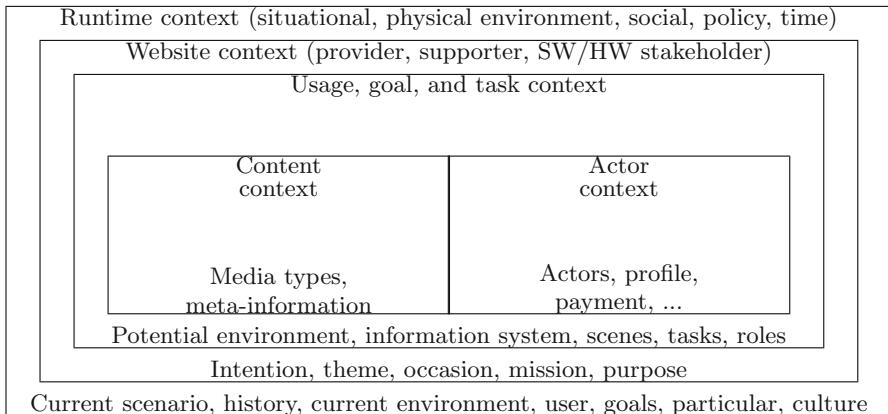


Fig. 2.5. Layers of context

2.1.5 The Atmosphere of a WIS

While the brand, mission statement, utilisation space and utilisation portfolio aim at the characterisation of content, functionality and usage of the WIS in strategic terms, the atmosphere addresses the gestalt of the WIS, i.e., how the WIS should be configured. At the end the WIS will be implemented by and presented through web pages, which should convey a uniform impression to the WIS users.

Categories characterising the impression of pictures can be used such as energetic, romantic, elegant, refreshing, harmonic, or stimulating. Each of these categories will have implications on the choice of form and colour, which we will discuss in detail in Part IV. On the strategic level the choice of one of these categories corresponds to the question what impression the WIS shall convey. The question is, what atmosphere is best suited for the envisioned content and functionality.

Example 2.6. In Example 2.5 we characterised the application for loans as a matching problem. Choosing a harmonic presentation for the WIS would suggest the intention to find an “optimal” solution for the customer and the bank. Choosing a stimulating atmosphere might suggest encouraging the customer in his or her application. An elegant atmosphere of the WIS may be chosen to convey confidence to the customer, i.e., that his or her financial affairs will be dealt with in the best way.

In addition, the atmosphere of a WIS is concerned with the progression patterns for the tasks. These patterns reflect the logical connection of information revealed to the user during the execution of a task. We distinguish between the following *progression patterns*:

- A *circular* progression pattern is centered around a particular content item, the phases of which form the core of the content to be delivered. At each stage details are added.
- A *loop* progression pattern emphasises the iteration of content, each time taking a different perspective. This is similar to a circular pattern, however puts more emphasis on changes.
- An *incremental* progression pattern emphasises the development of several content items over time. At each stage some of the items may be completed.
- An *evolutionary* progression pattern emphasises the stages of the content items, in particular those that are used in the result of tasks. At each stage content item may still be incomplete.
- A *network* progression pattern emphasises the flexible treatment of content items during the development of tasks and the logical connection between various such objects.

Example 2.7. If we choose a circular progression pattern for a personal loan application, the presentation of information will be centered around the loan, each time gaining a clearer picture of the result. In Example 2.5 we explained that it would be a good idea to successively discard possible loan options. This is in accordance with circular progression.

An incremental progression pattern would emphasise the various components of a loan application, i.e., the customer data, the conditions, and the budget. Each of these components would be treated as a separate item. While this is common in many WISs that offer form-oriented access, it may not be the best choice, because it does not support well the interaction between a tentative choice of conditions, the calculation of repayment costs and the personal financial situation of the customer.

An evolutionary progression would be very similar to an incremental one. However, each component could be left incomplete. This would help with the problem of changing conditions.

A loop progression would be similar to a circular progression. The difference is that the circular progression emphasises more the narrowing of options, while a loop progression would permit returning to options that have already been discarded.

Finally, a network progression is again similar to an incremental one, but leaves much more flexibility, as fixed order of the components is presumed.

Progression patterns have a direct impact on the placement of content on web pages, thus on the tiling of pages and the mapping of content to the tiles. We will discuss such a mapping of patterns to web page grids in detail in Part IV.

2.2 Strategic Analysis

Let us now address guidelines for strategic modelling. As already indicated in Section 2.1, we will discuss linguistic analysis, communication analysis, and metaphors.

2.2.1 Linguistic Analysis Using Word Fields

Linguistic analysis addresses the logical connections between the tasks, the content and the goals of users using techniques that are applied to analyse natural language descriptions of users' activities. The major source of information used for developing information systems is a description of processes, which may be delivered verbally or in the form of texts. Linguistic analysis considers such natural language descriptions and uses them for analysing the activities of the users: What are these activities? Does the description indicate any sequencing or continuation? What data is needed for or used by the activities? What are the relationships between these data? As user activities can be described by verbs, we suggest analysing the corresponding word fields. Linguistic analysis as part of strategic WIS modelling was briefly discussed in [727, 726].

According to [309] a word is an abstract concept, which in order to become concrete needs a word form carrying the grammatical variants. Word fields are a much more general notion than word forms. Word fields combine different aspects:

- morphology, i.e., the forms of the written or spoken word,
- phonology, i.e., the sound of the spoken word,
- syntax, i.e., the construction of sentences using the word forms,
- semantics, i.e., the meaning(s) of the word, and
- pragmatics, i.e., the usage of the word in written or spoken language.

If we restrict ourselves to written language communication considering the phonology becomes obsolete. Therefore, we tacitly assume that verbally communicated information in a formalised language usage context such as business communication can also be delivered in textual form without loss of semantics. If we had to analyse everyday language, this would no longer be true.

A *word field* [472, 759] is a linguistic system in which similar words that describe the “same” basic sense and are used in the same context are combined to a common structure and data set.

In contrast to the common synonym dictionary, word fields define the possible and necessary actors, the actions and the context. Word fields can be used for verbs, nouns and adjectives. We focus on *verb fields* and extend the implementations of the *WordNet* dictionary for verbs. Word fields are well-specified for a number of verbs. For each verb field we can at least define the following abstract information:

- basic semen,
- context of usage,
- possible/necessary actors, actions and context, and
- the star or snowflake specification, which comprises
 - the *logical structure* specifying possible arguments,
 - the *semantic description* of relevant and irrelevant valences, and the association with semantic type (colour, space, time, etc.),
 - the *kernel semantics* describing the word field with semen, and
 - *example sentences* providing additional information usable for help desk systems.

Example 2.8. A typical example is the verb “inform”, for which we can differentiate between four major facets of the word field:

1. To bring/advise something to somebody's attention something:

The word field is semantically characterized by: objectivity; functional information; official information; explanation; to be associated with something in future; using different representational media and presenters; with a degree of extraction from open or hidden; variety of styles such as short content description, long pertinent explanation, long event-based description. Typical related verbs are: announce, notify, publicize, publish, explain, present, herald, report, describe, outline, portray, pass a message, and scout.

2. To bring something that is unknown or hidden to somebody's attention:

The word field has a variety of presentation varying from proving a statement, arguing on latent facts, or unintentionally stating away. The association to the sender's way of thinking and acting is more strict than within the first variant. It is more personalized. Typical synonyms or associated verbs are: explain, report, clarify, elucidate, prove, admit, confess, scout, and betray.

3. To react on events or statements:

The reaction may be positive or negative. Typical application patterns are negotiations and adjustments. Typical associated other word fields are: confirm, approve, dispute, negate, prohibit, notify, state, announce, and promise.

4. To make suggestions for future activities:

A business site or an advertisement site does not display information without intention to generate surplus value. The intention of displaying information within an entertainment site is often to attract readers to become involved. Typical related actions are described by words such as propose, register, and offer.

We can generalise the theory of word fields as shown in [Figure 2.6](#). Generalised word fields are significantly different from word forms that carry the grammatical variants [309].

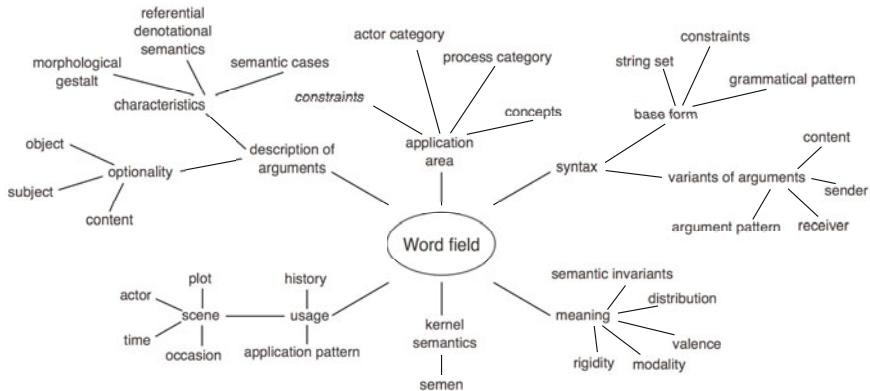


Fig. 2.6. The structure of a generalized word field

Under this premise we understand a *word field* to consist of its *intext*, its *context*, its *semantics* and its *pragmatics*.

- The *intext* combines morphology and syntax. We are mainly interested in the latter one, especially for verbs. In this case we may ask for the basic syntactical form, various mandatory and optional arguments, variants of the arguments, extensions to the basic syntax, and relationships and constraints between the arguments of the word when it is used in sentences. For instance, the verb “*to give*” uses a mandatory argument “*something*” and an optional argument “*to somebody*”. The basic syntactical form “*Someone gives something to somebody*” expresses an action, specifies the actor and the receiver, and an object used in the action.
- The *context* refers to application areas. According to the context we may identify related words, concepts related to the arguments, categories of actors and actions, and further constraints. For instance, in a banking context the verb “*to give*” may be used in a sentence such as “*The bank gives a home loan to the user*”. In this case we conclude that the object of the action is subject to contracted conditions regarding the use of the money for the claimed purpose, payment of principal and interest. A more precise statement would replace the verb “*to give*” by “*to grant*” or would replace the whole sentence by “*The bank offers a home loan contract to the user*”, in which case it would become clearer that a follow-on action, i.e., the acceptance of the offer, is expected.
- The *semantics* refers to the meaning of a word in a particular context. For example, the already used sentence “*The bank gives a home loan to the user*” includes the meaning of setting up a contract.
- The *pragmatics* refers to how the word is used in the application context.

Example 2.9. In electronic commerce verbs describing activities around contracts can to a large extent be considered as specialisations of the verbs “*to*

give”, “to take” and “to negotiate”. We will concentrate only on the first two of these, as the treatment of negotiations leads to a field in its own right.

As to the intext, the basic syntactic form associated with the verb “to give” is to give something to somebody. This indicates the existence of two roles, the giver and the taker, the involvement of an object, and an activity of the giver handing over the object to the taker. Of course, this also indicates a follow-on action with interchanged roles, which can be described by using the verb “take”.

The verb “to give” is quite abstract leaving enough space for interpretation, i.e., variants and constraints. For instance, the activity may be unconditional or conditional. In our example sentence “*The bank gives a home loan to the user*” the action definitely requires the follow-on acknowledgement to happen, whereas a sentence such as “*The company delivers the ordered goods within 10 days.*” — note that “*to deliver*” may be considered as a specialisation of “*to give*” — indicates an unconditional action. Further variants of “*to give*” are connected with specification of how the giving-action is to be executed. Unless the respective conditions are explicitly stated or generic business conditions apply, all variants are admissible. The verb itself does not express them.

The semantics of the verb “*to give*” depends on the context in which it is used. In a banking context it refers to the objects a bank can give to its users and vice versa, e.g., money, information, stock options, accounts, securities etc. In many cases the giving-action is coupled with choices and offers, which are not self-contained, but require follow-on actions. According to the more specific objects involved in the actions, more specialised verbs such as “*to offer*”, “*to inform*”, “*to grant*”, “*to propose*”, “*to present*”, “*to acknowledge*”, “*to confirm*” etc. are used. The related word fields for these verbs give more details on the semantics of a sentence in the give context.

As to the pragmatics, the use of the verb “*to give*” usually indicates that a follow-on take-action is assumed. Depending on the used specialisation, the continuation will also be specialised. For instance — staying within the banking context — the use of the verb “*to offer*” can be followed by “*to select*”, “*to choose*”, “*to accept*”, but also negated forms such as “*to reject*” or questioning conditions for taking are possible.

Example 2.10. Let us look again at loan applications focusing on the task of setting up a budget. This scene arises from a sentence such as “*The user and the bank set up a budget for the requested loan*”. Several word fields will be used in the analysis of this sentence. Central will be the verb “*to set up*”, which in general means to collect all the details describing the object argument, i.e., in this case “*budget*”. In the banking context this refers to the collection of information about debits and credits of the user including the payments for the loan.

This is determined by the word field of “*budget*”, which immediately leads to two components associated with the debits and the credits, respectively.

Furthermore, the word fields for “credit” and “debit” will indicate further specialisations, leading to the collection of information about salaries, rents, interests, rates, living expenses, etc.

2.2.2 Communication Analysis

Communication analysis addresses how a user will communicate with the WIS and why this communication is the best for the provider and the user. This helps to understand user goal and set up the utilisation portfolio accordingly. Communication analysis for WIS has been dealt with in detail in [397].

If a user can get the impression that guidance and information are as good as would be expected from a human, the system will more likely be accepted and thus be successful. From this we draw the conclusion that the WIS has to be defined in a way that best reflects the user-business communication in case there were no system support. Therefore, it appears to be a good idea to start with an analysis of this communication.

Communication analysis summarises the activities and techniques that are applied to understand the typical human-to-human interaction in the application domain, and classifies typical communication barriers. We use certain dimensions of understanding messages as a framework for understanding communication. These are used to identify communication flaws. We proceed with identifying different types of communication barriers, which affect the communication. Then we argue that using localisation abstraction and metaphors can help to overcome communication barriers, implying that their use may enhance the user’s understanding and successful navigation through a WIS.

Communication Dimensions

We may understand communication as an exchange of messages. Here we deal with semantic and pragmatic aspects of these messages. In particular, we are concerned with successful communication, i.e., communication by means of which both partners do meet their respective goals. We can distinguish the four main dimensions of understanding messages depicted in [Figure 2.7](#):

- *Content*, i.e., what the message is about;
- *Presentation*, i.e., what the sender tells about the message and the sender him- or herself;
- *Relationship*, i.e., what the sender thinks about the receiver and their relationship, and what the receiver thinks about the sender;
- *Appeal*, i.e., what the sender wants the receiver to do.

In WISs we mainly have to deal with written messages. These dimensions, however, apply to spoken messages, but can also be applied to written messages. Assuming that a necessary condition of non-successful communication is that at least one message is flawed with respect to one of the dimensions, we can distinguish the following message flaws:

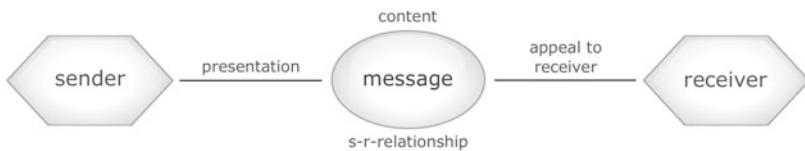


Fig. 2.7. Dimensions of understanding messages

- *Content flaw*, i.e., the sender does not say what he or she wanted to say, or the receiver understands the utterance differently;
- *Presentation flaw*, i.e., what the sender tells about him- or herself is inaccurate, or the receiver understands it incorrectly;
- *Relationship flaw*, i.e., what the sender thinks about his or her relationship to the receiver is incorrect, or the receiver understands it incorrectly;
- *Appeal flaw*, i.e., what the sender wants the receiver to do is not persuasive, or the receiver does not correctly understand the (valid) appeal issued by the sender.

Clearly the above mentioned dimensions of message understanding are not really independent in a formal sense and neither are the flaws. But as can be seen below, they help to formulate and classify communication barriers that appear to be realistic.

Improving Communication via Localisation Abstraction

Localisation abstraction addresses problems arising from content, presentation and relationship flaws. As the communication we are dealing with here is a mediated human-to-human interaction, various kinds of knowledge and ability have been represented inside the web information system as data or program. The information system in response to a user's inquiry chooses the data or program best suited. This requires having a model of the user (type) represented inside the web application. Given a particular inquiry and a user type, the most suited data or functionality available may be selected or constructed and the user given access to it. The web information system further obtains an answer to the inquiry that can be just the identified data as it is stored in its database. However, it can also be the result of applying business functionality to these data. Finally, the application actually delivers the answer to the user. Since the functionality required by the user is distributed over the information system's information space, the user might not be able to access it immediately, but might first need to position him- or herself on the most suitable location. Thus communication gets mixed up with navigation. Therefore, simplifying navigation can result in significantly reducing communication barriers. The simplification of choice is realised by supporting tools.

At least the following navigation functions appear worth supporting with tools:

- *Position signalling*, a function pointing out to the user his or her actual position in application space.
- *Heading determination*, a function determining the direction and also maybe the means best suited to approach a certain position.
- *Short distance environment exploration*, a function used to explore the immediate neighbourhood of a given location.
- *Long distance environment exploration*, a function used to explore the regions of the application space that are far away from a given location.

In a web information system these functions can be supported by the use of *localisation abstraction*. A particular view on the system's information space model is associated with a user and may be displayed to him or her. The user's location in the application space and his or her favourite locations may be part of this model. The model may be presented as a graph. Its vertices could represent locations and the edges the connections between them. The vertices could be labelled by the functionality, i.e., knowledge or ability that is available at the specific location. After each navigation step the model must be updated. Algorithms are then needed for obtaining, updating and displaying the models. It should be possible to obtain the needed results on the fly if the intended model is not too sophisticated.

Using the localisation abstraction properly can thus target navigation type barriers, as it reduces the perceived complexity of the information space and thus increases the users' ability to aim for his or her purpose. Reasonably chosen metaphors may help the users to apply their knowledge and abilities in a situation (i.e., while working on the information space) they are unfamiliar with. Using metaphors may cause users to be much more efficient in identifying the functionality required for their purpose, move to there, invoke the functionality and thus meet their goals more efficiently.

Example 2.11. Let us look at communication flaws in a system dealing with loan applications. Content flaws can be the following:

- Users have a variety of knowledge levels concerning loan types offered, their application areas, the terms and conditions, the assessment criteria, etc.
- Users may or may not have a clear picture about what the loan is needed or going to be used for. Some may have only a vague idea of the purpose.
- Users may depending on their level of education easily understand the differences between loan types and their respective suitability or not.

Presentation flaws in a loan application system can be as follows:

- Users may have a clear mind about their financial capability, and thus be able to set up a reasonable budget. Others may over- or under-estimate

their capability to afford the payments that will be required in case of application approval.

- Users may have sufficient computer literacy to handle an electronic system without difficulties or need a lot of help in doing so.
- Users might – due to their insufficient language fluency – need specific support while filling in the application form.
- Users might – due to their cultural or social background – not like to outline their financial details to the bank (or a computerised system).

Relationship and appeal flaws can take the following form:

- Users might – due to their personal background – tend to uncritically follow bank staff's advice.
- Users might – due to their cultural background, or family related reasons – not dare apply for a loan.
- Users might – due to their insufficient understanding of banking business or their difficult state of affairs – just appeal for help and overlook the fact that the bank only in rare cases can afford not to benefit from accepting a loan application.

2.2.3 Metaphors

Metaphorical structures, i.e., metaphors, allegories, metonymies or synecdoches, can be used to help the scanning or zapping user to understand a page or a website. Metaphorical structures are well understood in the linguistic community (see, for instance, the extensive selection of references in [186]; [38, 419, 632]). In this book we consider the following metaphorical structures: metaphors, allegory, metonymies and synecdoches [128, 646]. The utilization of metaphorical structures is based on structures of the brain and the cognition abilities [178, 419, 927]. Metaphors are known in the GUI community as an instrument to transfer the content of an icon or a button to the user. Except for [681], none of the texts on internet design such as [910, 784, 785] (or one of the other hundreds) treat them explicitly. The main usage of metaphors is still the simplified presentation of icons [650, 885]. The substitution direction of metaphorical structures has been developed by Aristotle [41]. [473] classifies metaphorical structures.

In general, metaphorical structures have a communicative or cognitive function [178, 187]. For example, many users do not realise that within a computer context a trash often stands as a metaphor for the action of deleting files. The interpretability of metaphorical structures depends on the common experience of the sender and the receiver. Metaphors such as *folder*, *files*, *trash can*, and *recycle bin* are common to computer users but not so common to naive users. However, metaphors can also mislead users, e.g., the use of a trash box for ejecting disks in an Apple environment or the use of a clipboard for singleton messages in a Windows environment. Naive users are not acquainted

with these metaphors. Thus, metaphorical structures, if they are to succeed, must be familiar to users. The users can understand content, functionality, and intention intuitively via using such metaphorical structures. WIS development can profit from well-integrated metaphorical structures that aim at focusing on a deeper context explanation as well as on a more specific navigation and selection.

Metaphorical structures can be used in the context of WIS design for supporting the scanning or zapping user and for the set-up of a story for the WIS. They can help introducing the content, style and “the feeling inside” of a complex web presentation. Mental models are also used in [807] to define persistent internet-metaphors: digital library, electronic mail, electronic marketplace, and digital world. These metaphors are based on ancient myths and archetypes that have influenced human thinking for thousands of years: keeper of knowledge (the digital library), communicator (electronic mail), the trader (electronic marketplace), and the adventurer (digital world). The internet representation of these metaphors is a matter of common knowledge, but the fact of using metaphors is mostly not known to users.

Metaphorical structures lead to more sophisticated user interfaces [379, 483] which are intuitively understandable, which can be used intuitively, whose “philosophy” or “mission” can be captured within seconds by novel users and need not be learned – a requirement sometimes called “grandma-safe”.

Metaphors should meet requirements such as simple interpretation, common sense of the meaning, abstraction in the right (user-centered) way and integrability into the dialogue [646]. In our daily language, metaphorical structures have great power, so why should we not use them within WISs? In order to do that, we have to integrate the metaphorical structures into the complete WIS development.

The summary of [477] in [447] states that metaphors refer to properties of concepts rather than words. Consequently their function is to help understand concepts rather than being just aesthetic or artistic expressions. Metaphors are an intrinsic and indispensable part of everyday language; they are easily used by “ordinary” people, and they are inevitable in human thought and reasoning. This is rephrased in [476] saying that the new view of metaphors takes imaginative aspects of reason – metaphor, metonymy, and mental imagery – as central to reason rather than as a peripheral and inconsequential adjunct to the literal. Thus, we may say that human communication is essentially metaphorical.

In general, a metaphorical structure [849] is the unusual usage of a language expression, i.e., using a language expression in a meaning that is not expected in the application context. The language expression is used as a language pictorial which works on the basis of a similarity of two objects/words. These objects/words are described by some dominant properties [130].

For metaphors there are two different definitions. According to the traditional definition a *metaphor* is characterised by a substitution of names. For instance, the notion “virus” stands for a “damaging computer program”.

A second, more modern definition characterises a *metaphor* as generating a new coherence of meanings. That is, two word fields are put into a similarity relation.

Example 2.12. Consider the sentence “URLSnoop is a Trojan Horse that gathers email addresses secretly”. In this case the characteristics of an event in ancient Greek mythology is projected onto the computer. In doing this the characteristics of the computer update the ancient myth leading to a modern serial of the Troy story.

An *allegory* is an extended metaphor that stands for a complex idea.

Example 2.13. “Orchestra” is used as an allegory for a group of people working together, e.g., a company. Related metaphors are the “conductor” for the executive manager of the company, the “first violin” for the star designer or the chief programmer, and “playing together” for working together. Thus, the allegory “orchestra” generates a complex field of several related metaphors.

A *metonymy* is a replacement of a term by something that is related to this term. The relation can be real, mental, factual, or causal.

Example 2.14. In the sentence “This Kafka is interesting” the author Kafka stands for one of his books. In “The Kieler will win the German League” the citizens of Kiel are standing for the Kiel handball team.

Synecdoches are closely related to metonymies. In contrast to a metonymy a synecdoche stands for a part-whole-relation. For instance, in the phrase “to start a computer” the notion “computer” stands for its operating system.

In addition to these four further metaphorical and rhetorical structures are known such as synonyms, paraphrases, analogies, oxymorons, emphases, or ironies.

The generation of metaphors could be supported by logically decomposing the information space into a small number of domains that appear homogeneous with respect to the offered functionality. This allows one to relate the domains to user types and expected user actions. Generation of metaphors appears then as being connected to finding characterizing names for the three valued relationships mentioned between user types, expected user action and domain. It might be useful for this to omit user related information from the user type that appears not to be relevant for the expected behavior. Further, methods from requirements engineering such as role-playing or brainstorming can be applied to generate first ideas on the metaphors to use. Clearly one must avoid, for a given user type, having too many conflicting or contradicting metaphors built into the web information system.

In order to use an appropriate linguistic representation for a metaphorical structure the following questions have to be considered:

- Who will the user be (experiences, age, knowledge etc.)?

- What kind of communication is used (verbal communications vs. written communication vs. internet communication etc.)?
- What are the possible communication objects (e.g., verbal communications have the gestures; the internet communication has the colours and icons)?
- Which connotations (i.e., social and emotional side conditions) should not be used?
- Which cultural background does the user have?

In general, metaphorical structures are used with different intentions. They can be predicative or persuading. Attributive, genitive, and apposition metaphorical structures add some information. Icons or colors illustrate properties. Thus the choice of a representation form is restricted. It is not possible that all cognitive characteristics can have all representation forms.

Metaphorical structures can be classified as follows:

- *conventional* metaphorical structures on the basis of stereotypes such as a phone icon;
- *ex-metaphorical structures* in technical terminology;
- *creative* metaphorical structures which require greater reasoning abilities from the user;
- *re-metaphorical structures* through activation of images with modification.

Example 2.15. Let us continue Example 2.11 and explore metaphors that could be used in loan applications. These metaphors correspond to message flaws to be dealt with below. These flaws are consequences of specific perceptions of the applicant itself, the bank and its staff. Since we are dealing with a quite formalized business we think these perceptions should be based on a conceptualization of this business. We identify three main roles (besides the decision-making) played by bank staff. The first role is that of an information provider, which leads to the *dictionary* metaphor. The second role is that of a helpful assistant who answers questions on how to fill in the form. This gives an *operations manual* metaphor. The third role is that of an adviser who provides advice about the most suitable loan type or the acceptability of the user for a particular loan. This gives an *adviser* metaphor.

The communication between a user applying for a loan and a bank employee dealing with loan applications is mainly determined by the metaphors described above, and some communication barriers. Recall that the identified metaphors mainly relate to the role of bank staff as seen by the user. To some degree this is a consequence of the fact that the loan application processing is highly rule based and only leaves a little latitude to bank staff.

Let us finally briefly consider communication barriers impacting the traditional loan application process. The level of education may be a barrier to successful communication between a user and bank staff. This is reflected in the user dimensions and should lead to further explanatory system components. Another severe communication barrier is the user's fear or bluff as its counterpart. A user might fear the loan application not being accepted,

especially if the loan is essential and the financial situation is tough. Communication barriers may also arise due to gender, age, ethnicity, cultural or family background.

Dealing with bluff and fear requires the system to be as transparent as possible and to emphasize the identified advice component. Inabilities with respect to language can be dealt with based on a multi language feasible design. Cultural background and ethnicity up to some degree can be reflected by the system supplying the applicable general legal regulations. Age and gender may be addressable by design and layout issues as well as focussing on the most required loan application areas and amounts.

2.3 Bibliographical Remarks

The chapter is based on the website development methods developed at Cottbus, e.g., [586]. The storyboarding approach generalises story development techniques [40, 208, 243, 772, 773, 887] used for movies etc. It can be enhanced agile development methods, e.g., [290].

Strategic modelling of software systems is based on [97, 314]. D. Bjørner [97] divides software engineering into three main phases: application domain description, requirement prescriptions, and system specifications. He calls these phases the *system development triptych*. The application domain description is a model describing the application domain, its entities, functions, events, and behaviour. It is based on a formal, semi-formal or natural language which allows to formulate a set of theorems or postulates or properties that are claimed to hold for the domain model. The triptych should be extended by architecture guidelines [359]. L. Heinrich [314] distinguishes design and development at the strategic layer, at the tactical layer and at the operational layer. Business informatics follows this approach.

Strategic WIS modelling has been considered as a specific form of requirements analysis for websites. We distinguish between the social system and the technical system. If this distinction is not made then a number of approaches apply: UML and object-oriented techniques for web engineering (e.g., [36, 129, 168, 169, 271, 283, 463, 684, 886, 889, 894, 909]), navigation-oriented techniques [167, 168, 169, 265, 284], web modelling language techniques and especially WebML [144] or the BPMN extension of WebML [906] with WebRatio, website design methods (e.g., WSDL [182, 183, 766], classical requirements development methods (e.g., [25, 337, 382, 654, 672]), and hypertext-backed methods (e.g., [269, 277, 761] or in combination with i* models [11, 246, 267, 275, 520]).

Generalising the notion [908, 693] in our terms, a mission is a collection of tasks assigned to a group or a person that is assigned by providers and owners of a website. The mission declares the stakeholders' core purpose and focus that normally remains unchanged over time. It defines goals, ethics, culture, and norms: what the website does for its customers, what it does for itself,

what it does for its owners, what the website does for its community, and what the website does for the world. It is based on goal models such as i* [18, 275, 935]. The vision is then derived from the mission.

Branding is a well-known thousand year old marketing concept that focuses on product identification or familiarity. The notion of business-to-business and business-to-consumer brand has already been introduced in the beginning of the web and has been extensively used since then, e.g., [445, 341]. We extended the provider-user characterisation to the provider-good-user-activity dimensions.

Utilisation portfolios [385, 740] combine approaches to portfolio analysis (e.g., [862]), information logistics (e.g., [188, 297, 905]), and task modelling (e.g., [630, 635]).

The utilisation context associates users and their utilisation in dependence on their devices and on current environment [406]. According to [926] we distinguish between the *intext* and the *context* of web content. Chapter 2 condenses the research on context [89, 401, 527, 823].

The atmosphere is an approach developed by stage developers for holistic design [580]. It can be derived from usability requirements [611], Gestalt thinking and psychology [434, 507, 517], and web design, e.g., [266, 777, 784, 785, 893, 910].

Word fields and especially verb word fields are developed in [427, 472, 759]. They are generalised to concept fields [202] or semantic fields and nets [329].

Communication analysis and interaction research (e.g., [78, 164, 309, 310, 316, 768, 767, 471]) is one of the central resources for user-friendly website design. Our approach is based on the experience obtained during development of natural language interface for database design [126, 127, 200]. It has been extended to design of dialogue systems [70, 74, 71, 73]. The meme approach [815] enhances our specific treatment of communication.

Metaphors associate some properties of a primary subject to another secondary subject as if they were one and the same, while the characteristics of the latter do not normally belong to the former. Aristotle [41] claims that “a metaphor is the application [to something] of a name belonging to something else”. Metaphorical structures provide a comfortable means to webpage design, e.g., [187, 419, 447, 476, 632, 807]. Metaphors are rudimentary rhetorical perception models [99, 855].

Key Messages

The **Strategic Modelling** of a web information system comprises

- a *mission statement* describing in general terms what the web information system is about;

- a *utilisation space* describing content, functionality and context in very general terms;
- a *utilisation portfolio* describing actors, goals and tasks in very general terms;
- a utilisation context addressing behaviour of users, supported environment and runtime support;
- a general description of the desired *ambience* of the WIS concerning layout and playout.

Strategic Modelling is supported

- by *linguistic analysis* centred around dependencies in word fields in connection with notions in the utilisation space and the utilisation portfolio;
- by *communication analysis* centred around the message exchange between users and the WIS;
- by *metaphors* associated with the communication of the users, cultural background and possible connotations.



Storyboarding

The goal of this chapter is to introduce *storyboarding* as the major activity on the business layer of web information systems. According to the traditional understanding of conceptual modelling storyboarding abstracts completely from implementation-minded system development. Nevertheless, for navigation within WISs this basic principle seems to be quite often forgotten, as many approaches specify navigation only on the level of web pages. It is very unlikely that such an approach could handle large, data-intensive WISs. Storyboarding on the other hand emphasises designing the story of the application instead. This approach has analogues in the areas of classical drama and movie production. Therefore, terms such as “story”, “scene”, “actor”, “plot”, etc. are used.

As WISs are open in the sense that anyone who has access to the web could become a user, the design of such systems requires some anticipation of the users’ behaviour. Storyboarding addresses this problem. Thus, a *Storyboard* will describe the ways users may choose to interact with the system.

A storyboard consists of three parts. The first section describes the paths users follow while navigating through the WIS. These paths are the *stories*, and their integrated description will be called the *story space*. We will deal with modelling the story space in Section 3.1. We will approach story space modelling first on a coarse level dealing with multi-layered labelled graphs, then obtain a finer granularity by using story algebras, for which we emphasise the language **SiteLang**. The story space model adds details to the utilisation space of the strategic layer that was discussed in Chapter 2.

The second part describes the *actors*, i.e., groups of users with the same behaviour. We will deal with modelling actors in Section 3.2. Actors are described by their information needs, which will lead to the modelling of information portfolios, the roles they can play in the system, and their characterising profiles. Roles are associated with intentions, i.e., what the actor wants to achieve from the system, rights, i.e., what the actor is permitted to do with the system, and obligations, i.e., what the actor is obliged to do. Profiles are based on the analysis of the various dimensions that impact on the behaviour

of the actor, especially on preferences on how to use the WIS. Actor modelling adds details to the utilisation portfolio of the strategic layer that was discussed in Chapter 2.

The third part describes *tasks*, which link the activities of the story space with the actors. We will deal with the modelling of tasks in Section 3.3. In particular, we will add details to the notion of task that was already introduced in Chapter 2 as part of the strategic WIS model. Finally, we wrap up our presentation in Section 3.4, where we emphasise again the overall picture of storyboarding.

As storyboarding is the core of the usage-oriented co-design approach to WIS modelling and development, it has been investigated deeply in a lot of preliminary work starting from a first description of stories in [222]. The gist of storyboarding was further developed in [747, 711, 529, 530] with applications to e-business [713, 721, 557], e-learning [721, 742], edutainment [736] and other categories of WIS, and finalised in [727, 726]. The contextual part of storyboarding was addressed in [399, 401]. Storyboarding also influenced the development of a theory of services in [748, 534], which was refined in [749].

3.1 Story Spaces

On a high level of abstraction we may think of a WIS as a set of abstract locations, which abstract from actual pages. A user navigates between these locations, and on this navigation path he or she executes a number of actions. We regard a location together with local actions, i.e., actions that do not change the location, as a unit called *scene*.

Then a WIS can be described by an edge-labelled directed multi-graph, in which the vertices represent the scenes, and the edges represent transitions between scenes. Each such transition may be labelled by an action executed by the user. If such a label is missing, the transition is due to a simple navigation link. The whole multi-graph is then called the *story space*.

Scenes may be atomic or complex. In the latter case the scene itself represents another set of abstract locations, between which users can navigate. This gives rise to a multi-layer model.

3.1.1 Scenario Modelling

Roughly speaking, a *story* is a path in the story space. It tells what a user of a particular type might do with the system.

The combination of different stories to a subgraph of the story space can be used to describe a “typical” use of the WIS for a particular task. Therefore, we call such a subgraph a *scenario* or *episode*. Usually storyboarding starts with modelling scenarios instead of stories, coupled by the integration of scenarios to the story space.

Definition 3.1. A *scenario* or *episode* \mathcal{E} consists of

- a finite set \mathcal{S} of *scenes*,
- an (optional) *start scene* $s_0 \in \mathcal{S}$,
- an (optional) set of *final scenes* $\mathcal{F} \subseteq \mathcal{S}$,
- a finite set \mathcal{A} of *actions*,
- a *scene assignment* $\sigma : \mathcal{A} \rightarrow \mathcal{S}$, i.e., each action α belongs to exactly one scene,
- and a *scene transition relation* $\tau \subseteq \mathcal{S} \times \mathcal{S} \times (\mathcal{A} \cup \{\text{skip}\})$, i.e., whenever there is a transition from scene $s_1 \in \mathcal{S}$ to scene $s_2 \in \mathcal{S}$, this transition is associated with an action $\alpha \in \mathcal{A}$ with $\sigma(\alpha) = s_1$ or with $\alpha = \text{skip}$, in which case it is a navigation without action, and we have $(s_1, s_2, \alpha) \in \tau$.

We write $\mathcal{E} = (\mathcal{S}, s_0, \mathcal{F}, \mathcal{A}, \sigma, \tau)$ or $\mathcal{E} = (\mathcal{S}, \mathcal{A}, \sigma, \tau)$, if there are no start and final scenes.

According to this definition each action $\alpha \in \mathcal{A}$ has a unique *source scene* $\sigma(\alpha) \in \mathcal{S}$. It also has a set $\bar{\tau}(\alpha) \subseteq \mathcal{S}$ of *target scenes*, which is obtained from the scene transition relation τ by

$$\bar{\tau}(\alpha) = \{s_2 \in \mathcal{S} \mid (\sigma(\alpha), s_2, \alpha) \in \tau\},$$

i.e., it contains all those scenes that can be reached by a scene transition labelled by α from the source scene of α . We may assume $\bar{\tau}(\alpha) \neq \emptyset$, otherwise the action would be useless and could be omitted from the scenario.

Another subtlety is that Definition 3.1 of a scenario does not exclude that the same action may be associated with more than one scene transition. That is, if a user chooses to execute a particular action α that is available in a scene s_1 , there may nevertheless be more than one possible successor scene s_2 . This reflects the fact that conditions that would determine the successor scene still are to be expressed at a finer level of detail, which is not yet present on the level of scenarios. We will take up this issue in Chapter 4 on a propositional level.

The fact that there may be no start scene for a scenario may appear a bit strange at first glance. However, it merely reflects the fact that scenarios are only sections of the story space. Technically, any such section can define a scenario. However, the story space as a whole must have a start scene. Analogously, if we refine a complex scene by a scenario, this scenario must also have a start scene. In other words, for scenes we request a well-defined entry point, whereas for scenarios this is not the case. Scenarios are used as a means to define the story space, while scenes are part of the story space that capture a particular functionality of the application. This will be reflected further on in the support for scenes and the specification of details, while none of this will be done for scenarios.

Graphical Representation of Scenarios

We can represent a scenario by an edge-labelled directed graph, in which the vertices correspond to the scenes, i.e., to \mathcal{S} with the start scene highlighted, and the edges to the scene transitions, i.e., to τ . [Figure 3.1](#) shows in general, how such a graphical representation of a scenario will look like.

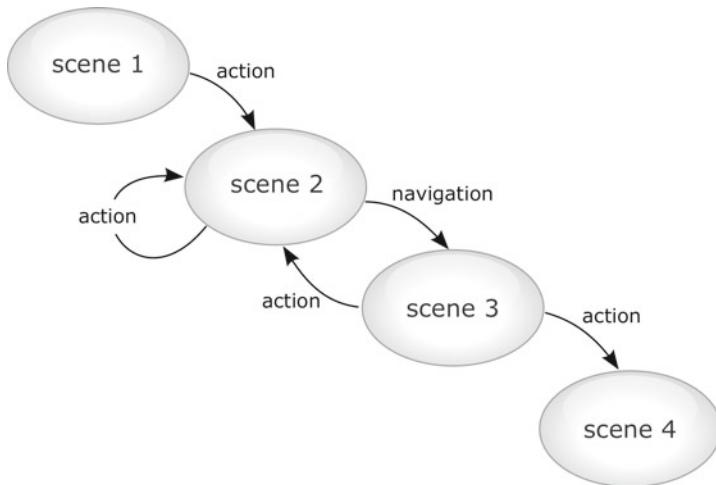


Fig. 3.1. Graphical representation of a scenario

A few more general remarks regarding scenarios are due here. A scenario is just a first sketch of the navigation of users through a WIS. It does not yet indicate which paths will be taken nor under which conditions certain paths will be followed. To that end a scenario just describes possible stories. In particular, the integration of all scenarios defines the story space, thus indicates all possible stories.

The complex scenes in a scenario – and thus also in the whole story space – request further scenarios for their specification. This leads to multiple layers of graphs, which we will explain in more details in Subsection 3.1.3.

Furthermore, a scenario does not indicate which users will select particular stories. This includes the specification whether users are entitled to execute certain actions or not as well as the question, which stories would be preferred by particular users.

Integration of Scenarios

An advantage of using scenarios is that they facilitate the integration of stories into the story space in a rather simple way. Assume two scenarios

$\mathcal{E}_i = (\mathcal{S}_i, s_0^i, \mathcal{F}_i, \mathcal{A}_i, \sigma_i, \tau_i)$ ($i = 1, 2$) have been set up. In order to integrate them, we first have to identify common scenes. That is, by renaming the scenes we may assume that we have

$$\mathcal{S}_1 = \{s_1, \dots, s_\ell\}$$

and

$$\mathcal{S}_2 = \{s_k, \dots, s_n\}$$

with $k \leq \ell + 1$, such that their union

$$\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 = \{s_1, \dots, s_n\}$$

can be taken as the set of scenes of the integrated scenario $\mathcal{E} = \mathcal{E}_1 \oplus \mathcal{E}_2$. If the start scenes s_0^1 and s_0^2 are the same in both scenarios, i.e., $s_0^1 = s_0^2 = s_i$ for some i with $k \leq i \leq \ell$, we take $s_0 = s_i$ as the start scene for the integrated scenario. Otherwise, the integrated scenario will not have a distinguished start scene. We define $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$ as the new set of final scenes.

In the same way we can treat the actions. Of course, we can only identify an action $\alpha \in \mathcal{A}_1$ with $\beta \in \mathcal{A}_2$, if we have $\sigma_1(\alpha) = \sigma_2(\beta)$. Assuming we can write

$$\mathcal{A}_1 = \{\alpha_1, \dots, \alpha_y\}$$

and

$$\mathcal{A}_2 = \{\alpha_x, \dots, \alpha_m\}$$

with $x \leq y + 1$, we obtain their union

$$\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2 = \{\alpha_1, \dots, \alpha_m\}$$

as the set of actions of the integrated scenario $\mathcal{E} = \mathcal{E}_1 \oplus \mathcal{E}_2$. We then get

$$\sigma(\alpha_i) = \begin{cases} \sigma_1(\alpha_i) & \text{if } i \leq y \\ \sigma_2(\alpha_i) & \text{else} \end{cases}$$

and $\tau = \tau_1 \cup \tau_2$ to complete the definition of the integrated scenario $\mathcal{E} = \mathcal{E}_1 \oplus \mathcal{E}_2 = (\mathcal{S}, s_0, \mathcal{F}, \mathcal{A}, \sigma, \tau)$.

The fact that scenario integration may lead to a loss of the start scene is a simple consequence of the slight arbitrariness of the notion of scenario. If scenarios are used to specify complex scenes and two such complex scenes are to be merged, then we must request that they use the same start scene.

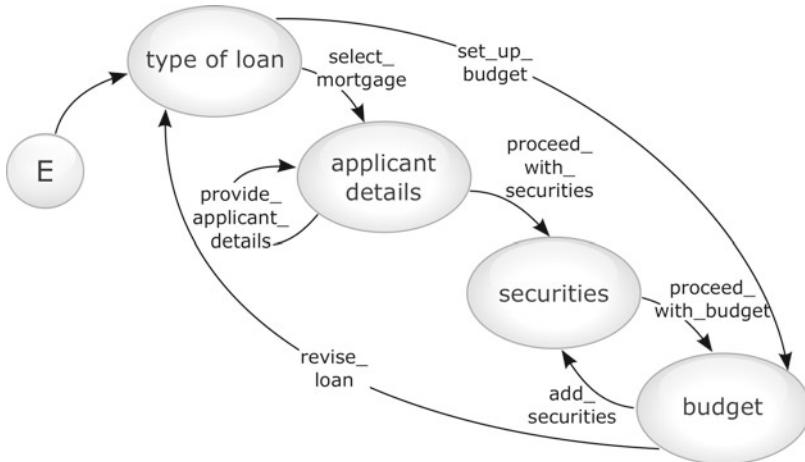


Fig. 3.2. Graph of a scenario for on-line loan applications

3.1.2 Examples in E-Business and E-Learning

Let us now look at an example in the area of electronic business. For this, we choose the area of electronic banking focussing on on-line loan applications and processing. A simplified version of this example has been used in [713, 712].

Example 3.2. The goal of such a web-based on-line loan system is to provide the loan best suiting the customer's purpose and to offer a loan in a way that the budget, i.e., payments in comparison to income, liabilities, etc., can be justified. Of course, loan applications can be rejected, e.g., if no satisfactory budget can be set up or if the securities are insufficient. However, the respective decisions nowadays are highly rule-based and do not leave much latitude to bank staff. Focussing only on the application side for an on-line loan, we can distinguish four major parts:

- The first part deals with the identification of *the loan type*. This type can be a mortgage for a house, a simple personal loan to purchase some goods, e.g., a car or furniture, an overdraft loan, etc. The customer is requested to select the type of loan, the amount of the loan including any extra costs, and to describe the purpose of using the loan.
- The second part deals with the *applicant details* and possible joint applicants, i.e., name, address, previous addresses if necessary, occupation, employer, contact details, etc. The requested information is normally the same for all applicants.
- The third part deals with *securities* offered to cover the loan, e.g., a dwelling, piece of land, life insurance, etc. If the security is owned by a third party, then an indication must be given that this party will guar-

antee the requested amount, i.e., that it acts as guarantor in the required manner.

- The fourth and largest part concerns setting up a *budget* for the customer. This contains all details about current income (salaries, rents, interest, etc.), commitments (existing loan payments, rates, living expenses, etc.), proposed payment for the requested loan, liabilities and assets.

Therefore, we can define a scenario with four scenes for each of these system parts. [Figure 3.2](#) shows a graphical representation of this scenario. However, actions have been omitted in this figure, as they are basically only navigation links. We will take a closer look at this example again in Section 3.1.5.

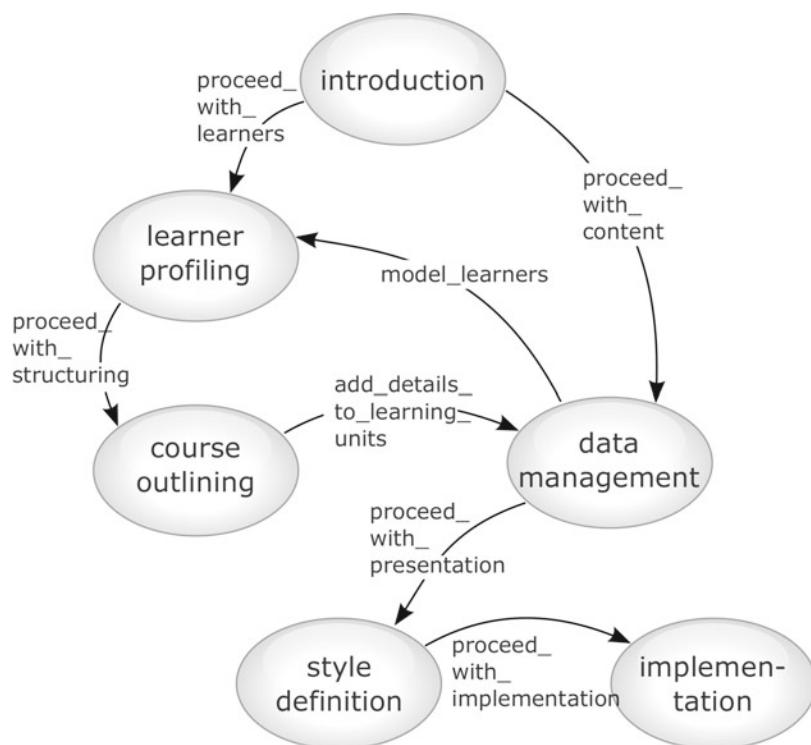


Fig. 3.3. Scenario for a course outline in e-learning

Let us now look at an example in the area of electronic learning. In the case of web-based learning systems a first design sketch may identify the scenes with *learning units*. Then the story space is also known as the *learning space*, and the graph of a scenario corresponds to an *outline graph* [90, 742].

Example 3.3. Take for example a course dealing with e-learning systems. Then we might have learning units such as Introduction, Learner Profiling, Course Outlining, Data Management, Adaptivity, Style Definition, and Implementation. [Figure 3.3](#) gives a rough picture of the corresponding course outline with links naturally represented by edges. Actions on a learning unit depend on the successful completion of the learning unit by the learner.

A simplified version of this example has been used in [742].

3.1.3 Adding Details to Actions and Scenes

At a finer level of detail we would like to extend the definition of a scenario. As already indicated, we would like to indicate, under which conditions an action or scene transition can or must be executed. We also want to indicate which actors appear in a scene and which actions they may execute. Further extensions concern the data content of a scene, the data produced by an action, and the data communicated by a scene transition.

Pre- and Postconditions

Therefore, we associate a *precondition* with each action $\alpha \in \mathcal{A}$ to specify exactly, under which conditions an action can be executed. Such preconditions can be easily expressed on the basis of propositional logic. That is, we take a set $\{\varphi_1, \dots, \varphi_n\}$ of atomic propositions for the scenario and complete this to a propositional logic defining the set \mathbb{F} of *propositional formulae* – or *propositions* for short – in the usual way, i.e.,

- Each atomic proposition φ_i is a proposition in \mathbb{F} .
- If $\varphi, \psi \in \mathbb{F}$, then also $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$ and $\varphi \rightarrow \psi$ are propositions in \mathbb{F} .

Thus, the *precondition* of an action $\alpha \in \mathcal{A}$ is a proposition $\text{Pre}(\alpha) \in \mathbb{F}$. A user can only choose the action α and execute it, if the precondition $\text{Pre}(\alpha)$ is satisfied in the current state of the WIS. In Section 3.1.5 we will look at such propositional preconditions in more detail. In Part III we will further refine such conditions linking them to the data content of the corresponding scene, i.e., we will leave the pure propositional level.

In a similar way we associate a *postcondition* $\text{Post}(\alpha) \in \mathbb{F}$ with each action $\alpha \in \mathcal{A}$ to specify exactly which effects the action will have. That is, if the action α is selected and executed, the resulting state of the WIS will satisfy the proposition $\text{Post}(\alpha)$. Same as with preconditions we pick up on propositional conditions on the story space in Section 3.1.5, and refine postconditions in Part III to conditions that depend on the data content of the story space.

Example 3.4. Take another look at Example 3.2, in particular at the action $\alpha = \text{select_mortgage}$. In this case we may want that a user can only execute this action, if he or she has received the necessary information about available

mortgages, i.e., mortgages must be known. This can be simply formalised by requesting $\text{Pre}(\text{select_mortgage}) = \text{mortgages_known}$ using an atomic proposition.

Analogously, we may set $\text{Post}(\text{select_mortgage}) = \text{mortgage_selected}$, which does not state more than that after executing α a mortgage has been selected.

While this may sound very simple at the moment, Example 3.15 will show in more detail, how pre-and postconditions of actions play a very important role in defining the action scheme of a story space, which will later become the important input for WIS personalisation.

Note that both pre- and postconditions are optional. Implicitly we always have true – also denoted as 1 – as a pre- and postcondition, which is satisfied in all states.

On a more global level, we may consider all actions α in one scene s , i.e., $\sigma(\alpha) = s$, such that α results in a scene transition, i.e., $(s, s', \alpha) \in \tau$ for some scene $s' \neq s$. If $\text{out}(s)$ denotes the set of all these actions, then taking the disjunction of all preconditions $\text{Pre}(\alpha)$ with $\alpha \in \text{out}(s)$ defines a necessary condition for leaving the scene s . We may strengthen this condition and define a sufficient *acceptance condition* for the scene s , denoted as $\text{Acc}(s)$. In particular, we must have

$$\models \text{Acc}(s) \rightarrow \bigvee_{\alpha \in \mathcal{A}, (s, s', \alpha) \in \tau \text{ for some } s' \neq s} \text{Pre}(\alpha).$$

Enabling and Triggering Events

In addition to a precondition the availability of an action to a user may also depend on an event. Therefore, we associate an *enabling event* with each action $\alpha \in \mathcal{A}$. This further clarifies the conditions, under which an action can be executed. Different from preconditions enabling events cannot be evaluated on the state of the WIS. They depend on the execution of other actions by the user or maybe other users as well. Formally, the enabling event can be specified by a list of actions that must have been executed already, and a list of actions that must not have been executed. In other words, the enabling event specifies whether a user is permitted to execute the action.

Analogously, we associate a *triggering event* with each action $\alpha \in \mathcal{A}$, which specifies whether a user is obliged to execute the action. Formally, a triggering event can also be specified by two lists of actions, those that must have been executed, and those that must not.

Example 3.5. Consider the action $\alpha = \text{proceed_with_securities}$ from Example 3.2. In this case the actions `select_mortgage` and `provide_applicant_details` must have been already executed. So these two actions will appear on the positive list specifying the enabling event of α , while the negative list is empty.

This means, a user who has selected a mortgage and entered the details of at least one applicant is permitted to enter securities for the mortgage.

The positive list of the specification of the triggering event for the action $\beta = \text{set_up_budget}$ from Example 3.2 contains the action `select_mortgage`, while the negative list is again empty. This means that if a user has selected a mortgage, he or she must set up a budget for it.

The specification of enabling and triggering events by positive and negative lists of actions is of course a rather coarse one. In a way these events prescribe rights and obligations of actors in a scene. Therefore, we will deal with this issue in Section 3.2.2 using a propositional deontic logic. In Part III we will further refine this logic and link it to the data content of the WIS.

Similar to the acceptance condition $\text{Acc}(s)$ for a scene we may also define an *acceptance event* using another pair of lists of actions. The positive list specifies which actions must have been executed, before the scene s can be left, while the negative list contains the actions that must not have been executed. We denote the acceptance event of scene s by $\text{acc_ev}(s)$.

Associated Actors

While events relate to obligations and rights of actors, there is a more direct association between scenes, actions and actors. That is, for each scene $s \in \mathcal{S}$ and each action $\alpha \in \mathcal{A}$ with $\sigma(\alpha) = s$ we would like to indicate which actors are likely to appear in that scene and execute that action. This amounts to associating with scenes and actions two more bits of information:

- We associate with scenes and actions a set of *roles* of actors indicating that only actors in these roles have access to the scene and can execute the action. Initially, we just use the role names that have already been identified on the strategic layer. In Section 3.2.2 we will deal with roles, which again will lead us to rights and obligations that can be expressed in a propositional deontic logic.
- We associate with scenes a set of *user types*, each of which can be described by certain characteristics. The user types capture which type of actors may appear at the scene and thus give information on how the scene should be designed. Again, we first use only indicative names for these user types. In Section 3.2.3 we will deal with user profiles and types including preference rules associated with them.

Information Portfolios and Manipulation Requests

Finally, we add details regarding the data that is processed in a scene. Firstly, with each scene we associate the data content that is presented to the user. We call this the *data consumption* of the scene. Actually, the data consumption of a scene $s \in \mathcal{S}$ should lead to a view V_s on some underlying database.

Therefore, we have to postpone a detailed discussion of data consumption to Part III. This will lead to a finer tuning of the story space. We may, however, already indicate which kind of data will be read by the scene. This is simply done by providing names of these data items.

Similar to the data consumption we may associate *data production* to each action $\alpha \in \mathcal{A}$, which has also be dealt with on a finer level of granularity, once operations that realise the actions can be defined. This will be covered later in Part III. However, we can already express the manipulation requests by indicating which data items will be written or updated.

Finally, each scene transition $(s_1, s_2, \alpha) \in \tau$ gives rise to data communication between scene s_1 and s_2 . On the level of scenarios we may add a name c_α for this data communication, i.e., we extend the scene transition to a quadruple $(s_1, s_2, \alpha, c_\alpha)$.

Example 3.6. For the scene type_of_loan in Example 3.2 we will have to read information about loans and loan samples. These define the data consumption for this scene. The action provide_applicant_details will write customer data, which defines the data production for this action. The data communication associated with the transition from scene applicant_details to securities consists of the selected mortgage and the applicants' details.

In all these cases we only provide a first indication of the data processed. Details will be given, when scene support is specified later on.

It would not make too much sense to add all this additional information to the graph that represents a scenario. In Section 3.1.7 we will use alternative representations for scenarios, which can be easier augmented by roles, user types, data consumption, production, and communication. However, pre- and postconditions and events are better captured on an algebraic or logical level in Sections 3.1.5 and 3.2.2.

3.1.4 Hierarchies of Scenes

So far we treated scenes in scenarios, as if they were atomic. Methodologically this would imply collecting scenarios, integrating them, and then proceeding with specifying the support details for each scene. However, we permit scenes to be complex, in which case they give rise to a scenario. That is, if s is a complex scene, there is a scenario $\mathcal{E}_s = (\mathcal{S}_s, s_{0,s}, \mathcal{F}_s, \mathcal{A}_s, \sigma_s, \tau_s)$, which specifies the details of the scene s . Note that we request that there is a start scene $s_{0,s}$ and final scenes \mathcal{F}_s . We call \mathcal{E}_s a *subscenario* of the scenario \mathcal{E} that contains the scene s .

Example 3.7. Figure 3.4 shows a subscenario for the scene type_of_loan in Example 3.2 handling the selection of loans. The rationale behind this sub-scenario is the following: It is likely that some users are not experts in the loan business. In particular, one can expect that some users do not know much about the offered spectrum of loan types. Customers lacking deep knowledge

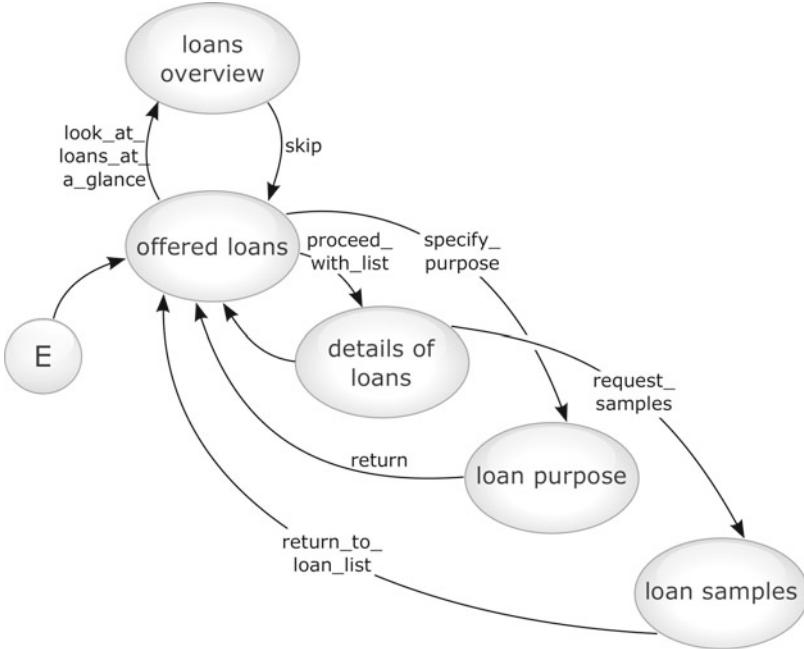


Fig. 3.4. Graph of a subscenario for the determination of a loan type

about loans may require in-depth explanations for each of the loan types including possible samples and loan conditions. As the loan business is a complicated one, it is likely that some customers will acquire knowledge of it while processing the application. Therefore, one may even expect that they change their initial goals.

We can now obtain a more detailed view by replacing the scene `type_of_loan` in Example 3.2 by this subscenario. Obviously, `offered_loans` would become the start scene of the integrated scenario, and the scene transition labelled by the action `revise_loan` now becomes a scene transition from `budget` to `offered_loans`. However, scene transitions that originally had `type_of_loan` as source scene have to be adapted as well, i.e., the transitions labelled by actions `set_up_budget` and `select_mortgage`, respectively, require a new source scene. In this example we simply choose this to be `offered_loans` in both cases. Figure 3.5 shows the result of this scenario expansion.

Example 3.7 suggests a general approach to replace a single complex scene s by its defining scenario \mathcal{E}_s .

Definition 3.8. Let $\mathcal{E} = (\mathcal{S}, s_0, \mathcal{F}, \mathcal{A}, \sigma, \tau)$ be a scenario and $s \in \mathcal{S}$ a distinguished scene in it. Furthermore, let $\mathcal{E}' = (\mathcal{S}', s'_0, \{s_f\}, \mathcal{A}', \sigma', \tau')$ be another scenario with $\mathcal{S}' \cap \mathcal{S} = \emptyset$. Then the *substitution of s in E by the subscenario E'* is the scenario $\mathcal{E}\{s/\mathcal{E}'\} = (\bar{\mathcal{S}}, \bar{s}_0, \bar{\mathcal{F}}, \bar{\mathcal{A}}, \bar{\sigma}, \bar{\tau})$ with:

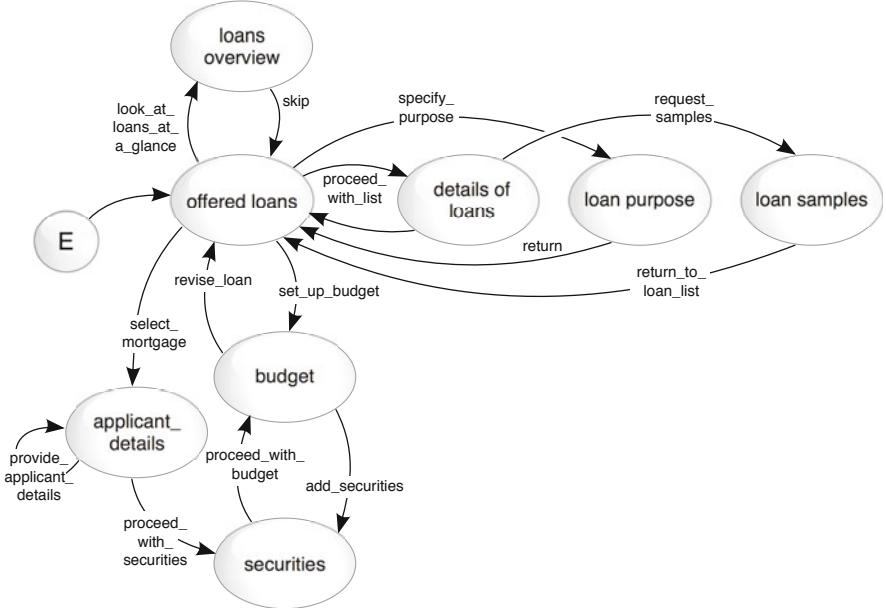


Fig. 3.5. Graph of an extended scenario for a loan application

$$\begin{aligned}
 \bar{\mathcal{S}} &= (\mathcal{S} - \{s\}) \cup \mathcal{S}' \\
 \bar{\mathcal{A}} &= \mathcal{A} \cup \mathcal{A}' \\
 \bar{s}_0 &= \begin{cases} s'_0 & \text{if } s = s_0 \\ s_0 & \text{else} \end{cases} \\
 \bar{\mathcal{F}} &= \mathcal{F} \\
 \bar{\sigma}(\alpha) &= \begin{cases} \sigma(\alpha) & \text{if } \alpha \in \mathcal{A}, \sigma(\alpha) \neq s \\ \sigma'(\alpha) & \text{if } \alpha \in \mathcal{A}' \\ s_f & \text{if } \alpha \in \mathcal{A}, \sigma(\alpha) = s \end{cases} \\
 \bar{\tau} &= \{(\bar{\sigma}(\alpha), s_2, \alpha) \mid (\sigma(\alpha), s_2, \alpha) \in \tau, s_2 \neq s\} \cup \\
 &\quad \{(\bar{\sigma}(\alpha), s'_0, \alpha) \mid (\sigma(\alpha), s, \alpha) \in \tau\} \cup \tau'
 \end{aligned}$$

In terms of the representing graph of a scenario we replace the scene s by the graph of \mathcal{E}' and redirect edges that start or end in s . Edges starting in s will be replaced by edges starting in some $s' \in \mathcal{S}'$, while edges ending in s will be replaced by edges ending in s'_0 , the start scene in \mathcal{E}' .

Note that the expansion of scenes in a scenario or the whole story space is only a way to provide a finer-grained presentation. However, it is sufficient to provide a scenario that defines the story space, then a lower level scenarios

for each non-atomic scene, etc. Further note that not each scenario represents a scene.

Example 3.9. Figure 3.6 shows a detailed scenario for loan applications, which may be considered as the result of a longer refinement process, in which further scenes in Figure 3.2 are expanded. In addition, loans are partitioned into personal loans and mortgages. Note that we numbered scenes and actions, respectively, so that we can easily refer to action α_i or scene s_j .

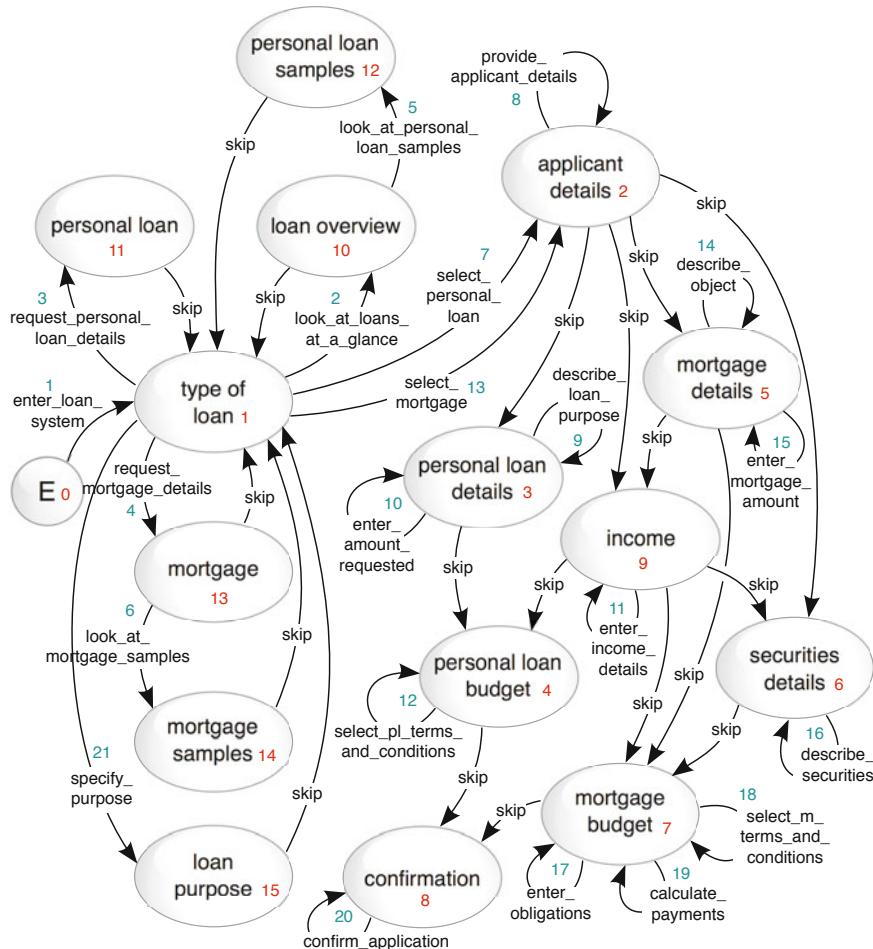


Fig. 3.6. A detailed scenario for on-line loan applications

The scenes provide important conceptual building blocks for the modelling of content and functionality at a local level. According to Section 3.1.3

an atomic scene is associated with a data consumption view plus the actions that can be executed on that view, i.e., a scene represents a *dialogue unit* in the sense of [716]. That is, a user may select some data from the scene, then select an action. Depending on the selected data and maybe additional input requested from the user this action initiates changes to the state of the WIS – we called this the data production – that are reflected in changes to some underlying database, and performs scene transitions. In this the identification of atomic scenes together with data consumption and production, manipulation requests and actions associated with the scene is the natural first step towards the definition of such dialogue units with details specified on a lower level of abstraction.

This view of scenes as defining means for dialogues is not restricted to atomic scenes. It also applies to complex scenes, in which case we do not just obtain a dialogue unit, but a complete representation of a dialogue between actor(s) and the WIS. The individual dialogue units that are associated with atomic scenes in this dialogue correspond to atomic dialogue steps. Actions that trigger transitions to and from a complex scene are external to the dialogue represented by the scene, whereas actions that appear in the defining scenario of the scene are internal.

So the modelling of scenes using detailed scenarios does not only contribute to setting up the complete story space. It also defines the frame for interaction of actors with the WIS through dialogues. For these dialogues the flow of actions is important, as it captures more details of WIS usage. We address this issue in the next Section 3.1.5.

3.1.5 Plots and Story Algebras

So far our presentation of story spaces was centered around the scenes as abstractions of locations between which users navigate and on which actions are selected and executed. Let us now look closer at the flow of actions within a scene or the story space. In both cases we obtain *action schemes* or *plots*, which refer to dialogues in the first case and to a detailed, yet still high-level specification of functionality in the second case.

Plots as Directed Graphs

So the first thing to do is to define the plot of a scene s assuming that we are given a defining scenario $\mathcal{E} = (\mathcal{S}, s_0, \mathcal{F}, \mathcal{A}, \sigma, \tau)$ for it. As we aim at the flow of actions, we turn \mathcal{A} into the vertex set of another directed graph. An edge between two actions indicates the sequencing of these actions. If there are several successors of an action, this may indicate a choice or parallel execution. In addition, we add an entry point and a leave point. This leads to the following definition.

Definition 3.10. Let $\mathcal{E} = (\mathcal{S}, s_0, \mathcal{F}, \mathcal{A}, \sigma, \tau)$ be a scenario. A *plot* \mathcal{P} for \mathcal{E} is a simple directed graph with vertex set $\mathcal{A} \cup \{E, L\}$ such that the in-degree of E and out-degree of L are both 0, and the following compatibility conditions are satisfied:

- There is at least one edge in \mathcal{P} from E to some $\alpha \in \mathcal{A}$ with $\sigma(\alpha) = s_0$.
- There is at least one edge in \mathcal{P} from some $\alpha \in \mathcal{A}$ to L .
- Whenever there is an edge from α to β in \mathcal{P} , we must have $(\sigma(\alpha), \sigma(\beta), \alpha) \in \tau$.
- For each $(\sigma(\alpha), s_2, \alpha) \in \tau$ there must exist an edge from α to some β with $\sigma(\beta) = s_2$ or to L .

We call E the *entry point* and L the *leave point* of the plot. If \mathcal{E} is the defining scenario of a scene s , then \mathcal{P} is called the *plot of scene s* . If \mathcal{E} defines a story space, then \mathcal{P} is called the *plot of the story space*.

Example 3.11. Figure 3.7 shows a plot for the scenario in Figure 3.5, which was explained in Example 3.7.

While scenarios for scenes and the story space provide a high-level application-oriented view on the WIS, plots provide an action-oriented view. These two views are complementary to each other. However, plots as defined in Definition 3.10 only give a glimpse of the flow of action. We still need more details.

In Section 3.1.3 we emphasised that we add enabling and triggering *events*, *preconditions* and *postconditions* to each action in a scenario to specify exactly, under which conditions an action can or must be executed and which effects it will have.

These extensions can then also available for the plots providing more details for the flow of actions. In addition, actions can be executed sequentially or in parallel, they can be iterated, and if several actions are available, users can choose between them. These extensions should also become part of the detailed specification of plots.

These possibilities to combine actions lead to operators of an algebra, which we will call a *story algebra*. Thus, we can describe a story space by an element of a suitable story algebra. We should, however, note already that story algebras have to be defined as being many-sorted in order to capture the association of actions with scenes.

The Language SiteLang

Let us take now a closer look at the storyboarding language **SiteLang** [850], which defines a story algebra that captures the details of plots. For this we take the set \mathcal{A} of actions and the set \mathcal{S} of scenes from a scenario (or the whole story space) \mathcal{E} as the basis for defining the set of *processes* $\mathcal{P} = \mathcal{P}(\mathcal{A}, \mathcal{S})$ determined by \mathcal{A} and \mathcal{S} . Furthermore, we can extend the scene assignment $\sigma : \mathcal{A} \rightarrow \mathcal{S}$ to a

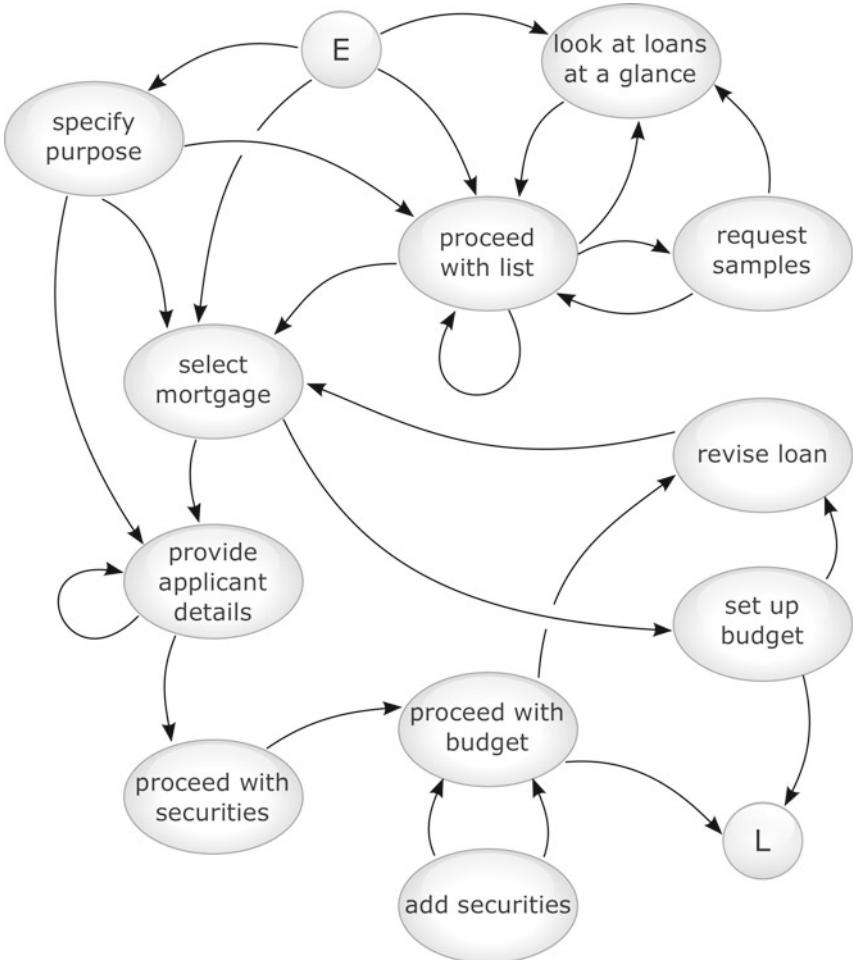


Fig. 3.7. Plot of an extended loan scenario

partial mapping $\sigma : \mathcal{P} \rightarrow \mathcal{S}$. Whenever $\sigma(p)$ is defined it denotes a unique scene to which the whole process p belongs, i.e., a user will not switch to a different scene while executing p . There are of course processes $p \in \mathcal{P}$ that span over more than one scene, in which case $\sigma(p)$ must be left undefined. Usually, the whole story space defines a process that does not belong to a unique scene.

We now present the syntax of **SiteLang**, but we deviate slightly from the syntax used in [850] and [89].

Definition 3.12. Let $\mathcal{E} = (\mathcal{S}, s_0, \mathcal{F}, \mathcal{A}, \sigma, \tau)$ be a scenario. The set of *processes* $\mathcal{P} = \mathcal{P}(\mathcal{A}, \mathcal{S})$ determined by \mathcal{A} and \mathcal{S} and the extension of the scene assignment σ to a partial mapping $\sigma : \mathcal{P} \rightarrow \mathcal{S}$ are inductively defined as follows:

- Each action $\alpha \in \mathcal{A}$ is also a process, i.e., $\alpha \in \mathcal{P}$, and the associated scene $\sigma(\alpha)$ is already given.
- **skip** is a process, for which $\sigma(\text{skip})$ is undefined.
- If p_1 and p_2 are processes, then also the *sequence* $p_1; p_2$ is a process. Furthermore, if $\sigma(p_1) = \sigma(p_2) = s$ or one of the p_i is **skip**, while for the other p_j we have $\sigma(p_j) = s$, then $\sigma(p_1; p_2)$ is also defined and equals s , otherwise it is undefined.
- If p_1 and p_2 are processes, then also the *parallel process* $p_1 \| p_2$ is a process. Furthermore, if $\sigma(p_1) = \sigma(p_2) = s$ or one of the p_i is **skip**, while for the other p_j we have $\sigma(p_j) = s$, then $\sigma(p_1 \| p_2)$ is also defined and equals s , otherwise it is undefined.
- If p_1 and p_2 are processes, then also the *choice* $p_1 \square p_2$ is a process. Furthermore, if $\sigma(p_1) = \sigma(p_2) = s$ or one of the p_i is **skip**, while for the other p_j we have $\sigma(p_j) = s$, then $\sigma(p_1 \square p_2)$ is also defined and equals s , otherwise it is undefined.
- If p is a process, then also the *iteration* p^* is a process with $\sigma(p^*) = \sigma(p)$, if $\sigma(p)$ is defined.
- If p is a process and φ is a boolean condition, then the *guarded process* $\{\varphi\}p$ and the *post-guarded process* $p\{\varphi\}$ are processes with $\sigma(\{\varphi\}p) = \sigma(p\{\varphi\}) = \sigma(p)$, if $\sigma(p)$ is defined.

SiteLang provides some few more constructs, which can be considered as syntactic sugar. Constructs such as non-empty iteration p^+ and optionality $[p]$ can be expressed by the constructs above, as we have $p^+ = p; p^*$ and $[p] = p \square \text{skip}$.

Furthermore, **SiteLang** provides constructors \nearrow and \searrow to enter or leave a scene, respectively. The use of these constructors is optional, as we may simply use parentheses and make the associated scene explicit in the definition of σ . If e.g., we had a process

$$p = (p_1; p_2) \square (p_1; p_3) = p_1; (p_2 \square p_3)$$

with $\sigma(p_1) = \sigma(p_2) = s_1$ and $\sigma(p_3) = s_2 \neq s_1$, then $\sigma(p)$ would be undefined. However, we also have $\sigma(p_1; p_2) = s_1$, so we could write

$$p = \nearrow p_1; p_2 \searrow \square (p_1; p_3) = p_1; (p_2 \square p_3).$$

SiteLang also uses $\parallel \varphi$ to mark a parallel execution with a synchronisation condition φ , which can be expressed by a post-guarded parallel process $(\dots \| \dots)\{\varphi\}$.

Finally, **SiteLang** foresees the possibility to specify a minimum and maximum number of processes that have to be chosen from a list of alternatives. That is, for given processes p_1, \dots, p_k we may specify a process

$$p = p_1 \square^{(m,n)} \dots \square^{(m,n)} p_k$$

with the informal meaning that out of the processes p_1, \dots, p_k at least m and at most n processes, say $p_{i_1}, \dots, p_{i_\ell}$ with $m \leq \ell \leq n$ and $1 \leq i_1 \leq \dots \leq i_\ell \leq k$,

must be chosen and executed in parallel. So, this operator $\square^{(m,n)}$ is just an abbreviation for a large choice

$$\dots \square (p_{i_1} \| \dots \| p_{i_\ell}) \square \dots$$

running over all i_1, \dots, i_ℓ with $m \leq \ell \leq n$ and $1 \leq i_1 \leq \dots \leq i_\ell \leq k$.

If the scenario $\mathcal{E} = (\mathcal{S}, s_0, \mathcal{A}, \sigma, \tau)$ in Definition 3.12 specifies a scene s or the whole story space, it is possible to associate a single process $p \in \mathcal{P}$ with it. We regard this process definition as the detailed specification of the plot of scene s or the story space, respectively. It provides the most detailed action-oriented view. We will now see how to define this plot.

Compatibility of Processes and Scene Transitions

According to Definition 3.12 we tacitly ignore navigation between scenes, as a process already specifies possible successor scenes. However, the scene transition relation τ in the scenario \mathcal{E} does also specify transitions, but τ was not used in the definition. Therefore, we have to identify those processes $p \in \mathcal{P}$ that are compatible with the scene transition relation τ of the given scenario. We do this by identifying the set of stories that are admitted by a process $p \in \mathcal{P}$.

Definition 3.13. Let $\mathcal{E} = (\mathcal{S}, s_0, \mathcal{F}, \mathcal{A}, \sigma, \tau)$ be a scenario and let $p \in \mathcal{P}(\mathcal{A}, \mathcal{S})$ be a process defined over \mathcal{E} .

1. A *story* over \mathcal{S}_C is a sequence $(\alpha_1, \dots, \alpha_k)$ such that for all $i = 1, \dots, k$ either $\alpha_i \in \mathcal{A}$ or $\alpha_i = \varphi$ with a boolean condition φ holds.
2. The *set st(p) of stories admitted by p* is inductively defined as follows:
 - For $p = \alpha \in \mathcal{A}$ the only story in $st(\alpha)$ is (α) .
 - For $p = \text{skip}$ the only story in $st(\text{skip})$ is the empty story $()$.
 - For a sequence $p = p_1; p_2$ each story in $st(p)$ has the form $(\alpha_1, \dots, \alpha_k, \beta_1, \dots, \beta_\ell)$ with $(\alpha_1, \dots, \alpha_k) \in st(p_1)$ and $(\beta_1, \dots, \beta_\ell) \in st(p_2)$ such that $(\sigma(\alpha_k), \sigma(\beta_1), \alpha_k) \in \tau$ holds, i.e., $st(p_1; p_2)$ contains the concatenations of a story admitted by p_1 and a story admitted by p_2 .
 - For a parallel process $p_1 \| p_2$ each story in $st(p_1 \| p_2)$ results from interleaving a story $(\alpha_1, \dots, \alpha_k) \in st(p_1)$ with a story $(\beta_1, \dots, \beta_\ell) \in st(p_2)$, i.e., the story has the form $(\gamma_1, \dots, \gamma_{k+\ell})$ with $\{\gamma_1, \dots, \gamma_{k+\ell}\} = \{\alpha_1, \dots, \alpha_k\} \cup \{\beta_1, \dots, \beta_\ell\}$ and
 - for $\gamma_i = \alpha_{i'}$, $\gamma_j = \alpha_{j'}$ with $i < j$ we must have $i' < j'$, and
 - for $\gamma_i = \beta_{i'}$, $\gamma_j = \beta_{j'}$ with $i < j$ we must have $i' < j'$.
 - For a choice $p_1 \square p_2$ each story in $st(p_1 \square p_2)$ is a story $(\alpha_1, \dots, \alpha_k) \in st(p_1)$ or a story $(\beta_1, \dots, \beta_\ell) \in st(p_2)$, i.e., $st(p_1 \square p_2) = st(p_1) \cup st(p_2)$.
 - For an iteration p^* each story in $st(p^*)$ has the form $(\alpha_{11}, \dots, \alpha_{1n_1}, \dots, \alpha_{k1}, \dots, \alpha_{kn_k})$ with stories $(\alpha_{i1}, \dots, \alpha_{in_i}) \in st(p)$ and any $k \in \mathbb{N}$ including 0.

- For a guarded process $\{\varphi\}p$ each story in $st(\{\varphi\}p)$ has the form $(\varphi, \alpha_1, \dots, \alpha_k)$ with $(\alpha_1, \dots, \alpha_k) \in st(p)$.
 - For a post-guarded process $p\{\varphi\}$ each story in $st(p\{\varphi\})$ has the form $(\alpha_1, \dots, \alpha_k, \varphi)$ with $(\alpha_1, \dots, \alpha_k) \in st(p)$.
3. The process p is *compatible* with the scene transition relation τ iff for all stories $(\alpha_1, \dots, \alpha_k) \in st(p)$ admitted by p and all $i = 1, \dots, k - 1$ the following holds:
 If α_i is an action, i.e., $\alpha_i \in \mathcal{A}$, and α_j is the next action in the story, i.e., $j > i$ and all $\alpha_{i+1}, \dots, \alpha_{j-1}$ are boolean conditions, then we must have either $(s_1, s_2, \alpha_i) \in \tau$ for $s_2 = \sigma(\alpha_j)$ or $(s_1, s_2, \alpha_i) \in \tau$, $\sigma(\alpha_j) = s_3$ with $(s_2, s_3, \text{skip}) \in \tau$.

Given a scenario $\mathcal{E} = (\mathcal{S}, s_0, \mathcal{F}, \mathcal{A}, \sigma, \tau)$, we may refine it by choosing a process $p \in \mathcal{P}(\mathcal{A}, \mathcal{S})$ that is compatible with τ as its detailed specification. This is the plot specification mentioned above. In particular, this applies to the whole story space. We then call such a p a *story space expression*. However, by abuse of notation we also refer to p simply as the story space.

3.1.6 Examples of Plots

Let us now look at examples for story space expressions. We will first use a very simple story space for ordering products, then continue the more complicated example dealing with loan applications.

Example 3.14. First take a very simple example, where the WIS is used for ordering products. In this case we may define four scenes:

- The scene $s_0 = \text{product}$ would contain product descriptions and allow the user to select products.
- The scene $s_1 = \text{payment}$ will be used to inform the user about payment method options and allow the user to select the appropriate payment method.
- The scene $s_2 = \text{address}$ will be used to require information about the shipping address from the user.
- Finally, scene $s_3 = \text{confirmation}$ will be used to get the user to confirm the order and the payment and shipping details.

There are six actions (their names are sufficient to indicate what they are supposed to do):

- $\alpha_1 = \text{select_product}$ is defined on s_0 and leads to a transition to scene s_1 .
- $\alpha_2 = \text{payment_by_card}$ is defined on s_1 and leads to a transition to scene s_2 .
- $\alpha_3 = \text{payment_by_bank_transfer}$ is defined on s_1 and leads to a transition to scene s_2 .
- $\alpha_4 = \text{payment_by_cheque}$ is defined on s_1 and leads to a transition to scene s_2 .

- $\alpha_5 = \text{enter_address}$ and is defined on s_2 and leads to a transition to scene s_3 .
- $\alpha_6 = \text{confirm_order}$, is defined on s_3 and leads to a transition to scene s_0 .

Finally, we get also five Boolean conditions:

- The condition $\varphi_1 = \text{price_in_range}$ expresses that the price of the selected product(s) lies within the range of acceptance of credit card payment. It is a precondition for action α_2 .
- The condition $\varphi_2 = \text{payment_by_credit_card}$ expresses that the user has selected the option to pay by credit card.
- The condition $\varphi_3 = \text{payment_by_bank_transfer}$ expresses that the user has selected the option to pay by bank transfer.
- The condition $\varphi_4 = \text{payment_by_cheque}$ expresses that the user has selected the option to pay by cheque.
- The condition $\varphi_5 = \text{order_confirmed}$ expresses that the user has confirmed the order.

Using these actions and conditions, we can formalise a story space by the algebraic expression

$$(\alpha_1; (\{\varphi_1\}\alpha_2\{\varphi_2\} \square \alpha_3\{\varphi_3\} \square \alpha_4\{\varphi_4\}); \alpha_5; (\alpha_6\{\varphi_5\} \square \text{skip}) \square \text{skip})^*.$$

Example 3.15. A story space expression for the detailed loan application in Example 3.9 can be described as follows:

```

enter_loan_system ;
(( {\varphi_0} (specify_purpose {\varphi_{13}} \square skip) ; look_at_loans_at_a_glance
  \square ( {\varphi_1} request_personal_loan_details ;
    ( look_at_personal_loan_samples \square skip ) {\varphi_3} ) \square
    ( {\varphi_2} request_mortgage_details ;
      ( look_at_mortgage_samples \square skip ) {\varphi_4} ) )^* {\varphi_5} ) ;
( select_personal_loan {\varphi_6} \square select_mortgage {\varphi_7} ) ;
( ( {\varphi_6} ( provide_applicant_details ;
  ( provide_applicant_details \square skip ) ;
  ( describe_loan_purpose || enter_amount_requested || enter_income_details ) ;
  select_pl_terms_and_conditions ) {\varphi_8} ) \square
  ( {\varphi_7} ( provide_applicant_details ; provide_applicant_details* ;
    ( describe_object || enter_mortgage_amount || describe_securities* ) ;
    ( enter_income_details || enter Obligations* ) ;
    ( ( {-\varphi_{12}} select_m_terms_and_conditions ;
      calculate_payments )* ;
      { \varphi_{12}} select_m_terms_and_conditions ) ) {\varphi_9} ) ) ) ;
confirm_application {\varphi_{10} \vee \varphi_{11}}

```

involving the set of scenes $\mathcal{S} = \{s_0, \dots, s_{15}\}$ with

$s_0 = \text{start}$	$s_1 = \text{type_of_loan}$
$s_2 = \text{applicant_details}$	$s_3 = \text{personal_loan_details}$
$s_4 = \text{personal_loan_budget}$	$s_5 = \text{mortgage_details}$
$s_6 = \text{securities_details}$	$s_7 = \text{mortgage_budget}$
$s_8 = \text{confirmation}$	$s_9 = \text{income}$
$s_{10} = \text{loan_overview}$	$s_{11} = \text{personal_loan}$
$s_{12} = \text{personal_loan_samples}$	$s_{13} = \text{mortgage}$
$s_{14} = \text{mortgage_samples}$	$s_{15} = \text{loan_purpose}$

The set of actions is $\mathcal{A} = \{\alpha_1, \dots, \alpha_{21}\}$ using

$\alpha_1 = \text{enter_loan_system}$	$\alpha_2 = \text{look_at_loans_at_a_glance}$
$\alpha_3 = \text{request_personal_loan_details}$	$\alpha_4 = \text{request_mortgage_details}$
$\alpha_5 = \text{look_at_personal_loan_samples}$	$\alpha_6 = \text{look_at_mortgage_samples}$
$\alpha_7 = \text{select_personal_loan}$	$\alpha_8 = \text{provide_applicant_details}$
$\alpha_9 = \text{describe_loan_purpose}$	$\alpha_{10} = \text{enter_amount_requested}$
$\alpha_{11} = \text{enter_income_details}$	
$\alpha_{12} = \text{select_pl_terms_and_conditions}$	$\alpha_{13} = \text{select_mortgage}$
$\alpha_{14} = \text{describe_object}$	$\alpha_{15} = \text{enter_mortgage_amount}$
$\alpha_{16} = \text{describe_securities}$	$\alpha_{17} = \text{enter_obligations}$
$\alpha_{18} = \text{select_m_terms_and_conditions}$	$\alpha_{19} = \text{calculate_payments}$
$\alpha_{20} = \text{confirm_application}$	$\alpha_{21} = \text{specify_purpose}$

The story space expression involves the Boolean conditions

$\varphi_0 \equiv \text{information_about_loan_types_needed}$
$\varphi_1 \equiv \text{information_about_personal_loans_needed}$
$\varphi_2 \equiv \text{information_about_mortgages_needed}$
$\varphi_3 \equiv \text{personal_loans_known}$
$\varphi_4 \equiv \text{mortgages_known}$
$\varphi_5 \equiv \text{available_loans_known}$
$\varphi_6 \equiv \text{personal_loan_selected}$
$\varphi_7 \equiv \text{mortgage_selected}$
$\varphi_8 \equiv \text{personal_loan_application_completed}$

$$\begin{aligned}
\varphi_9 &\equiv \text{mortgage_application_completed} \\
\varphi_{10} &\equiv \text{applied_for_personal_loan} \\
\varphi_{11} &\equiv \text{applied_for_mortgage} \\
\varphi_{12} &\equiv \text{payment_options_clear} \\
\varphi_{13} &\equiv \text{loans_recommended}
\end{aligned}$$

Finally, we get the scene assignment σ with

$$\begin{array}{lllll}
\sigma(\alpha_1) = s_0 & \sigma(\alpha_2) = s_{10} & \sigma(\alpha_3) = s_{11} & \sigma(\alpha_4) = s_{13} & \sigma(\alpha_5) = s_{12} \\
\sigma(\alpha_6) = s_{14} & \sigma(\alpha_7) = s_1 & \sigma(\alpha_8) = s_2 & \sigma(\alpha_9) = s_3 & \sigma(\alpha_{10}) = s_3 \\
\sigma(\alpha_{11}) = s_9 & \sigma(\alpha_{12}) = s_4 & \sigma(\alpha_{13}) = s_1 & \sigma(\alpha_{14}) = s_5 & \sigma(\alpha_{15}) = s_5 \\
\sigma(\alpha_{16}) = s_6 & \sigma(\alpha_{17}) = s_7 & \sigma(\alpha_{18}) = s_7 & \sigma(\alpha_{19}) = s_7 & \sigma(\alpha_{20}) = s_8 \\
\sigma(\alpha_{21}) = s_1
\end{array}$$

The process in this example is compatible with this scenario represented in [Figure 3.6](#). For instance, the story

$$\alpha_1 \varphi_0 \alpha_{21} \varphi_{13} \varphi_0 \alpha_2 \varphi_1 \alpha_3 \varphi_3 \varphi_5 \alpha_7 \varphi_6 \alpha_8 \alpha_8 \alpha_{10} \alpha_9 \alpha_{11} \alpha_{12} \varphi_8 \alpha_{20} \varphi_{10}$$

expresses the behaviour of a user who first enters the purpose of his or her requested loan (α_{21}) to receive some recommendations (φ_{13}), then looks at loans at a glance (α_2), requests details on a personal loan (α_3), selects a personal loan (α_7), enters details for two applicants (α_8), enters the amount requested (α_{10}), the purpose of the loan (α_9), and income details (α_{11}), selects terms and conditions (α_{12}), and finally confirms the application (α_{20}). However, the scenario does not give information about the flow of actions such as whether actions are executed in parallel or sequentially or whether they are iterated or optional, whereas the story space expression above does.

3.1.7 Alternative Representations for Scenarios and Plots

The graphical representation of a scenario does not leave much room for adding roles, user profiles, communicated data or any of the other extensions discussed in Section 3.1.3. In order to facilitate such extensions we may prefer using alternative representations such as *communication matrices*, a textual *Storyboard description catalogue* or *statecharts* [304, 306]. We will now present these alternative representations and discuss their merits.

Communication Matrices

Let $\mathcal{E} = (\mathcal{S}, s_0, \mathcal{F}, \mathcal{A}, \sigma, \tau)$ be a scenario. If we want to focus on the transition and communication between scenes, we may replace the scene transition relation τ by a *scene transition function*

$$\tilde{\tau} : \mathcal{S} \times \mathcal{S} \rightarrow \mathfrak{P}(\mathcal{A}) ,$$

where \mathfrak{P} denotes the powerset constructor. That is, we associate with each pair (s_1, s_2) of scenes the set $\tilde{\tau}(s_1, s_2)$ of actions that can be executed on scene s_1 and will result in a transition to scene s_2 .

When using such a scene transition function we may not only associate actions with a transition, but extend it by the communicated data. We obtain a function

$$\tilde{\tau} : \mathcal{S} \times \mathcal{S} \rightarrow \mathfrak{P}(\mathcal{A} \times \mathcal{T}) ,$$

where \mathcal{T} denotes a set of data types. This captures the data communication from Section 3.1.3. We retain the notion *scene transition function* for this extended function. Data types will be formally introduced later. For the moment let it just be a name describing informally, which data is to be communicated between scenes.

In the same way we could add roles and user types to scene transitions. However, it is advantageous to associate roles and user types with the scenes instead of the transitions, which leads to a *scene description function*

$$\lambda : \mathcal{S} \rightarrow \mathfrak{P}(\mathcal{R}) \times \mathfrak{P}(\mathcal{U}) ,$$

where \mathcal{R} denotes the set of roles, and \mathcal{U} the set of user types. Then $\lambda(s) = (\{r_1, \dots, r_k\}, \{u_1, \dots, u_\ell\})$ means that only actors in the roles r_1, \dots, r_k can access the scene s , and only actors with one of the user types u_1, \dots, u_ℓ are likely to appear at that scene.

In order to capture also data consumption and production – as described in Section 3.1.3 – we assume a set \mathcal{D} of *abstract data stores*. Each action can read from a data store $d \in \mathcal{D}$, update it, or write to it. This gives rise to another function

$$\mu : \mathcal{S} \rightarrow \mathfrak{P}(\{r, u, w\} \times \mathcal{D}) ,$$

which we call the *manipulation request function*. That is, for each scene s a pair $(\vartheta, d) \in \mu(s)$ indicates a manipulation request at that scene. For $\vartheta = r$ this is a read-request, i.e., data from the abstract data store d will only be consumed. For $\vartheta = w$ we obtain a write-request, i.e., data will only be produced and stored in the abstract data store d . In case $\vartheta = u$ we have an update-request, i.e., data from the abstract data store d will be consumed, and other data will be produced and stored in d .

This definition of manipulation requests needs further refinement on lower levels of abstraction. Finally, the data stores that are used for data consumption should be replaced by views on some database schema, while those used for data production should be replaced by the database schema itself. This will lead to a set of (extended) views, which we call *media types* in Part III, i.e., the data content, functionality and context of the scene will be supported by the media type M .

Furthermore, if the role r is associated with the scene s , we may also assume that in an abstract sense r communicates with the scene s , i.e., we

can use data types $\gamma_p(r, s), \gamma_c(s, r) \in \mathcal{T}$ to model this communication. The first of these data types denotes the information produced by an actor in this role, the second one the information communicated back by the scene and consumed by the actor.

Summarising these modifications and extension to the definition of a scenario, we obtain the following:

Definition 3.16. A *communication scenario* \mathcal{E} consists of

- a finite set \mathcal{S} of *scenes*,
- an (optional) *start scene* $s_0 \in \mathcal{S}$,
- a finite set \mathcal{A} of *actions*,
- a *scene assignment* $\sigma : \mathcal{A} \rightarrow \mathcal{S}$, i.e., each action belongs to exactly one scene,
- a set \mathcal{R} of *roles*,
- a set \mathcal{U} of *user types*,
- a set \mathcal{T} of *data types*,
- a *scene transition function* $\tilde{\tau} : \mathcal{S} \times \mathcal{S} \rightarrow \mathfrak{P}(\mathcal{A} \times \mathcal{T})$,
- a *scene description function* $\lambda : \mathcal{S} \rightarrow \mathfrak{P}(\mathcal{R}) \times \mathfrak{P}(\mathcal{U})$,
- a set \mathcal{D} of *abstract data stores*,
- a *manipulation request function* $\mu : \mathcal{S} \rightarrow \mathfrak{P}(\{r, u, w\} \times \mathcal{D})$,
- and *interaction functions* $\gamma_p : \mathcal{R} \times \mathcal{S} \rightarrow \mathcal{T}$ as well as $\gamma_c : \mathcal{S} \times \mathcal{R} \rightarrow \mathcal{T}$.

We write $\mathcal{E} = (\mathcal{S}, s_0, \mathcal{A}, \sigma, \mathcal{R}, \mathcal{U}, \mathcal{T}, \tilde{\tau}, \lambda, \mathcal{D}, \mu, \gamma_p, \gamma_c)$.

If we further neglect actions within a scene, i.e., we ignore all $\tilde{\tau}(s, s)$ with $s \in \mathcal{S}$, a communication scenario can be easily represented by a matrix as indicated in general in [Figure 3.8](#).

The core part of such a *communication matrix* is a (n, n) -matrix, where n is the number of scenes. These scenes are placed on the diagonal of the submatrix with the start scene in the uppermost position. The content of the other cells is determined by $\tilde{\tau}$ in the obvious way, i.e., $\tilde{\tau}(s_i, s_j)$ is placed in the (i, j) -cell of the matrix.

We may then extend the matrix in a way that treats roles in the same way as scenes. In particular, the data types $\gamma_p(r_i, s_j)$ and $\gamma_c(s_j, r_i)$ are placed in the (i, j) - and (j, i) -cells, respectively, of the extended matrix.

Finally, the matrix is extended to the right by the user types associated with the scenes via λ , and to the bottom by the manipulation requests that are specified by μ .

Though it is intuitively clear how to construct communication matrices, we can also give a formal definition for them.

Definition 3.17. Let $\mathcal{E} = (\mathcal{S}, s_0, \mathcal{A}, \sigma, \mathcal{R}, \mathcal{U}, \mathcal{T}, \tilde{\tau}, \lambda, \mathcal{D}, \mu, \gamma_p, \gamma_c)$ be a communication scenario with $\mathcal{R} = \{r_1, \dots, r_k\}$, $\mathcal{S} = \{s_1, \dots, s_\ell\}$, $\mathcal{U} = \{u_1, \dots, u_m\}$ and $\mathcal{D} = \{d_1, \dots, d_n\}$. The *communication matrix* associated with \mathcal{E} is a $(k + \ell + n, k + \ell + m)$ -matrix C defined as follows:

Role1		comm.data R1 -> Sc1			
	Role2	comm.data R2 -> Sc1			
		Scene1		comm.data Sc1 -> Sc3	
comm.data Sc2 -> R1			Scene2		comm.data Sc2 -> Sc4
			comm.data Sc3 -> Sc2	Scene3	comm.data Sc3 -> Sc4
comm.data Sc4 -> R1	comm.data Sc4 -> R2				Scene4
		read	update		
		write		update	read
					d1
					d2

Fig. 3.8. Matrix representation of a communication scenario

$$C(i, j) = \begin{cases} r_i & \text{for } 1 \leq i = j \leq k \\ s_{i-k} & \text{for } k + 1 \leq i = j \leq k + \ell \\ \tilde{\tau}(s_{i-k}, s_{j-k}) & \text{for } k + 1 \leq i, j \leq k + \ell \text{ and } i \neq j \\ \gamma_p(r_i, s_{j-k}) & \text{for } 1 \leq i \leq k, k + 1 \leq j \leq k + \ell \\ \gamma_c(s_{i-k}, r_j) & \text{for } 1 \leq j \leq k, k + 1 \leq i \leq k + \ell \\ u_{j-k-\ell} & \text{for } k + 1 \leq i \leq k + \ell, k + \ell + 1 \leq j \leq k + \ell + m, \\ & \lambda(s_{i-k}) = (R, U) \text{ and } u_{j-k-\ell} \in U \\ d_{i-k-\ell} & \text{for } k + \ell + 1 \leq i \leq k + \ell + n \text{ and } j = k + \ell + 1 \\ \vartheta & \text{for } k + \ell + 1 \leq i \leq k + \ell + n, k + 1 \leq j \leq k + \ell \\ & \text{and } (\vartheta, d_{i-k-\ell}) \in \mu(s_{j-k}) \\ \text{undefined} & \text{else} \end{cases}$$

Example 3.18. Figure 3.9 shows a communication matrix for the scenario from Example 3.2.

Storyboard Description Catalogue

The storyboard description catalogue provides a textual way to specify the story space, i.e., to capture scenarios, plots and all the added details. The story space is described by a scenario, the *main scenario*. Then each other scenario is described in the following way:

Customer	loan request loan purpose	customer data	offered securities	income liabilities assets	
loan description sample	type of loan		selected loan information	selected loan information	all user types
	applicant details			applicant data	all user types
security request			securities		all user types
risk assessment decision	risk assessment			budget	all user types
	read			update	loans
	write			update	customers
		write			securities
				write	payments

Fig. 3.9. Communication matrix for an on-line loan application scenario

Scenario:	\langle name of the scenario \rangle
Scenes:	\langle list of scene names \rangle
Start Scene:	\langle scene name \rangle
Final Scenes:	\langle list of scene names \rangle
Actions:	\langle list of action names \rangle
Transitions:	\langle list of transitions \rangle
Process Expression:	\langle SiteLang process \rangle

We can use the name “main” for the scenario that defines the whole story space. In this case we replace **Process Expression** by **Story Space Expression**. The start scene is optional. Each transition has the form
from \langle scene name \rangle **to** \langle scene name \rangle **via** \langle action name \rangle **using** \langle data type \rangle ,
but here the last part defining the data communication is also optional.

For scenes and actions we obtain further entries in the catalogue. Each scene is described in the following way:

Scene:	\langle scene name \rangle
Actions:	\langle list of action names \rangle
Roles:	\langle list of role specifications \rangle
User Types:	\langle list of user type names \rangle
Defining Scenario:	\langle scenario name \rangle
Acceptance Condition:	\langle condition \rangle

The actions for scene s are those actions α , for which we have $\sigma(\alpha) = s$ in the scenario that defines s . Each role specification takes the form

`<role name> consuming <data type> producing <data type>`,

thus capturing also the communication between the roles and the scene. The defining scenario is optional – it is only required for complex scenes.

Finally, for actions we use the following description:

Action:	\langle action name \rangle
Precondition:	\langle condition \rangle
Postcondition:	\langle condition \rangle
Roles:	\langle list of role names \rangle
User Types:	\langle list of user type names \rangle
Enabling Event:	+ \langle list of actions \rangle - \langle list of actions \rangle
Triggering Event:	+ \langle list of actions \rangle - \langle list of actions \rangle
Manipulation Requests:	\langle list of manipulation requests \rangle

Here, each manipulation request has the form $\vartheta \langle$ data store \rangle , where ϑ is either “read”, “write” or “update”.

In Section 3.2 we look at the modelling of actors. This may give rise to further entries in the storyboard catalogue, in particular for roles and user types.

Example 3.19. Let us look at our very first example of a scenario, the loan application scenario in Example 3.2 that was illustrated in [Figure 3.2](#). In this case the description of the story space would look as follows:

Scenario:	main
Scenes:	type_of_loan, applicant_details, securities, budget
Start Scene:	type_of_loan
Actions:	select_mortgage, provide_applicant_details, set_up_budget, proceed_with_securities, add_securities, proceed_with_budget, revise_loan
Transitions:	from type_of_loan to applicant_details via select_mortgage, from type_of_loan to budget via set_up_budget, from securities to budget via proceed_with_budget, from applicant_details to applicant_details via provide_applicant_details, from applicant_details to securities via proceed_with_securities, from budget to securities via add_securities, from budget to type_of_loan via revise_loan

We omitted the story space expression, which can be added later. For each of the four scenes we now obtain a detailed description in the form described above. One of these scene descriptions would look as follows:

Scene:	type_of_loan
Actions:	select_mortgage, set_up_budget
Roles:	customer consuming loans producing loan_request
Defining Scenario:	loan_information

In this scene description we do not yet provide user types nor an acceptance condition. These will be added later. The defining scenario loan_information is the one illustrated in [Figure 3.4](#). We omit its description in the storyboard catalogue.

Finally, we have to describe the seven actions in the main scenario. The following is the description for select_mortgage:

Action:	select_mortgage
Precondition:	loans_known
Postcondition:	mortgage_selected
Roles:	customer

In this case user types, triggering and enabling event and manipulation requests have been omitted.

The advantage of the storyboard description catalogue is that it can be easily extended and checked for completeness. In particular, we may add informal explanations and working comments. Later on these informal explanations can be replaced or complemented by exact formal definitions.

Statecharts

Statecharts were originally introduced by Harel to visualise complex dynamic systems [304]. Basically, a statechart consists of states and events. An event is a transition between states that is described by a condition and an action. So, in comparison to scenarios we basically have the same ground model of a directed labelled multi-graph, where the states can be considered as scenes and vice versa, and the transitions correspond to actions. In both cases we can build a hierarchical model as described above for subscenarios. So we could use statecharts to visualise scenarios, complete story spaces, and plots.

Example 3.20. Consider the scenario in [Figure 3.10](#), which is based on the detailed scenario in Example 3.9. We split the scenario into two complex scenes, which we named loan_information_and_selection and application_details.

[Figure 3.11](#) shows how statecharts can be used to capture the details of SiteLang expressions. The statechart specifies application_details using a hierarchy of scenes. In particular, the complex scene dealing with mortgages

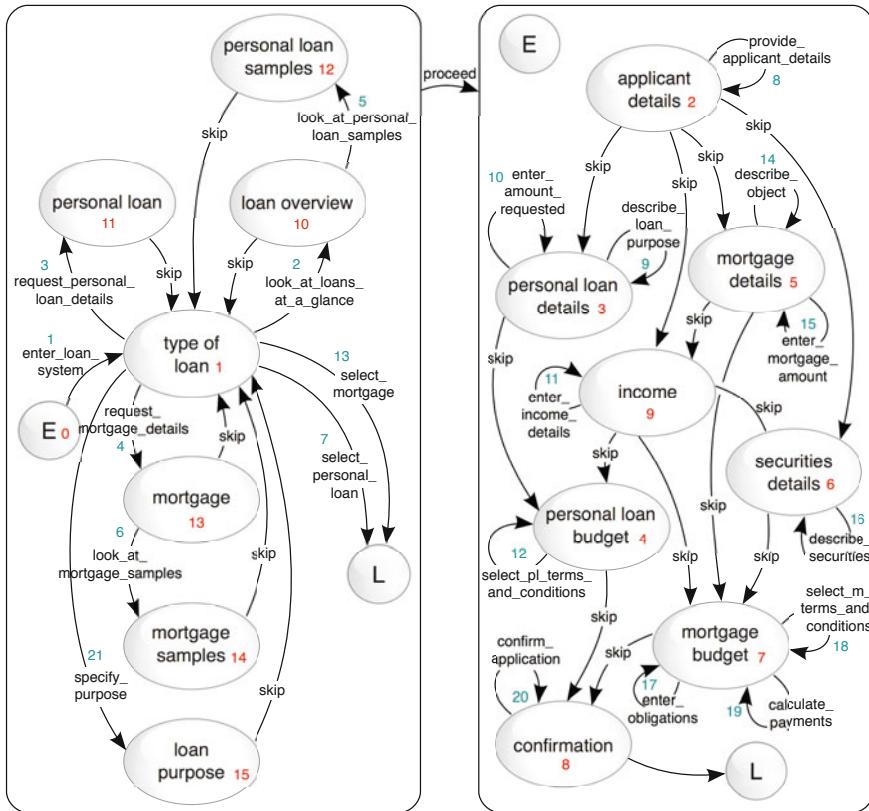


Fig. 3.10. Complex scenes in a loan application story space

uses three scenes income, mortgage_details and security_details that are accessed in parallel. Analogously, the complex scene dealing with personal loans uses two scenes personal_loan_details and income that are accessed in parallel. So the complex scene income appears in two different scenarios.

In addition, we added a history state to this statechart indicating that a user may return at any time to loan_information_and_selection without being forced to redo all actions on the scene application_details.

Note that statecharts can represent the process extensions to scenarios, i.e., they combine in a way scenarios with their plots. Nevertheless, roles, user profiles, rights, obligations and intentions are not captured. While statecharts provide a highly expressive modelling instrument, they have to be used with care.

Statecharts may contain details that we do not yet want to appear on the high level of abstraction, on which storyboarding resides. For instance, we have to keep in mind that scenes are not web-pages, whereas in approaches

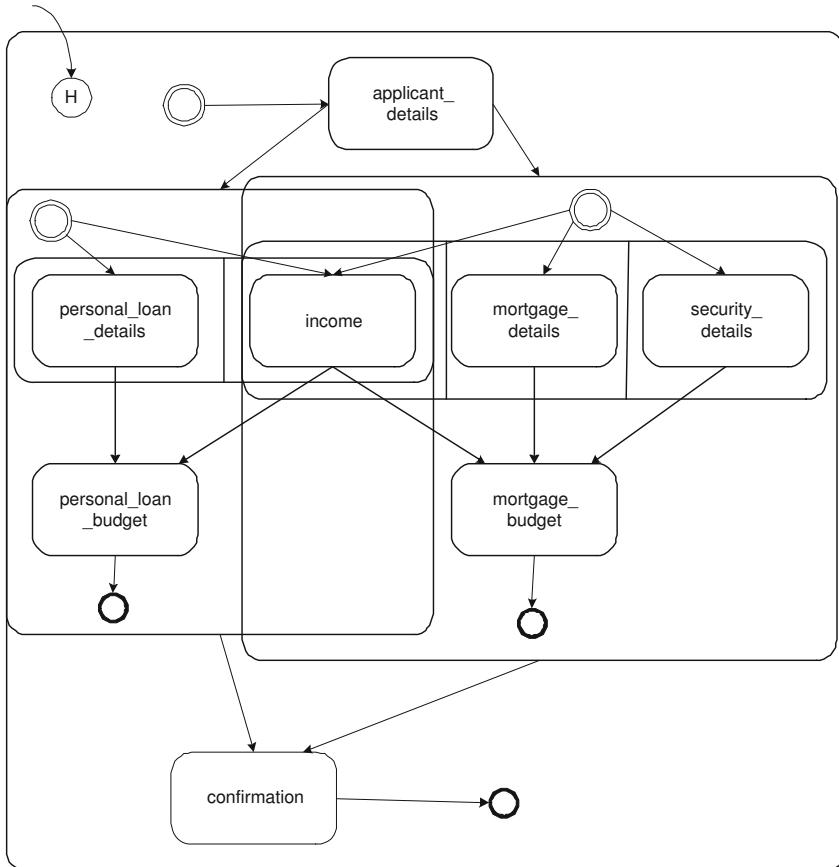


Fig. 3.11. Statechart involving parallel actions

like StateWebCharts [923] – which claims to extend statecharts – the states are indeed identified with pages leading to the unavoidable problem that the complexity of systems with several thousand pages cannot be handled properly on this level.

3.2 Actor Modelling

The story space mainly explains on a high level of abstraction, what a WIS is about and how it can be used. In order to complete the picture we also need to know who will be using the WIS, and how it will be used. That is we turn to the users of the WIS.

Users can be classified according to their roles, goals and behaviour. We use the term *actor* for such a group of users. The *role* of an actor indicates a

particular purpose of the system. As such it is usually associated with obligations and rights, which lead to deontic integrity constraints. Roles are also connected with the *tasks* that were identified on the strategic layer, as discussed in Chapter 2. Tasks will be handled separately in Section 3.3. The *goals* of an actor can be modelled by postconditions to the story space. Modelling the behaviour of an actor leads to *user profiles*, which can be modelled by giving values for various properties that characterize a user. Furthermore, each profile leads to preference rules that can again be expressed by constraints on the story space.

In addition, each actor has an *information portfolio*, which specifies the information needs as well as the information entered into the system. We do not model the information portfolio as part of an actor, but as part of the information consumption and production of scenes, which were discussed in Section 3.1.

3.2.1 Information Portfolios

Each WIS user who enters the system with a particular goal has information needs that have to be satisfied by the system. In addition, an active WIS will also request information from its users. We use the term *information consumption* for the information provided by the system to its users, and *information production* for the information entered by a user into the system.

When a user enters the WIS, the information needs are usually not known in advance. Part of the needed information may depend on other parts, on decisions made while navigating through the WIS, and even on the information provided by the actor him- or herself. That is, the information consumption and production depends on the path through the WIS, i.e., in our terminology on the story. Therefore, we associate information consumption and production with each scene of the story space. Assuming that there is a database for the data content of the WIS with database schema \mathcal{S} , information consumption on a scene s definitely accounts for a *view* V_s over \mathcal{S} . That is, we have another schema \mathcal{S}_V and a computable transformation from databases over \mathcal{S} to databases over \mathcal{S}_V . Such a transformation is usually expressed by a query q_V . Such views will form the basis of the theory of media types, which we will treat intensively in Part III.

However, a little subtlety comes in here. Views $V_s = (\mathcal{S}_V, q_V)$ are defined in a way that applying them to a database db over the schema \mathcal{S} results in a database $q_V(db)$ over \mathcal{S}_V , i.e., in sets. If we assume that $q_V(db)$ is just one set of objects, then the information that is made available to an actor corresponds to only one object in this set. This object is determined by parameters provided by the actor, i.e., by the information production on the previous scene of the story. This further implies that the information production should be associated not just with a scene, but with an action at that scene.

Thus, we obtain the following extensions to story spaces:

- With each scene s we associate a view $V_s = (\mathcal{S}_V, q_V)$ called *information consumption view*. Elements of $q_V(db)$ for some database db represent the *information consumption* of an actor.
- With each action α we associate a data type t_α called *information production type*. Values of type t_α represent the *information production* by an actor.

Information consumption and information production of an actor for all scenes together define the information portfolio of the actor.

Example 3.21. Take another look at the scene $s_1 = \text{type_of_loan}$ in Example 3.15. In this case the information consumption of any actor is a list of loans. In scene $s_{10} = \text{loan_overview}$ this would be extended by a short explanation for each of them. For the scene $s_{13} = \text{mortgage}$ the information consumption of an actor would be the description of one particular type of mortgage. This implies that action $\alpha_4 = \text{request_mortgage_details}$ needs this particular type of mortgage to be selected by the actor. Thus, the data type describing mortgages would be used to define the information production associated with α_4 .

In Part III we will take a closer and more formal look at information production and consumption in the context of media types.

3.2.2 Roles, Rights and Obligations

Roles are used to classify actors according to their rights and obligations. For instance, in a web-based conference system we may have roles for the programme committee chair(s), the programme committee members, and for authors. In an on-line loan system we may distinguish between actors in the role of customers and those in the role of bank clerks. In most systems we may expect one default role that is taken on by any actors, whereas other roles require some form of identification or at least an active switch to this role.

A *role* is defined by the set of actions that an actor with this role may execute. In Section 3.1.3 we already associated with each scene and each action in the story space a set of role names, i.e., whenever an actor comes across a particular scene, he or she will have to have one of these roles. In doing so, we can obtain a role-specific, i.e., *personalised* view of a scenario. If r is a role, discard all scenes and actions in a scenario that do not contain r in their sets of associated roles, then also discard all transitions for which source or target scene has been deleted.

Analogously, we may obtain role-specific plots. Each action that does not have the role r in its associated set of roles, will simply be replaced by `fail`. We can then apply axioms for processes to simplify the resulting plots.

This way of discarding actions is only a rather coarse way to obtain role-specific scenarios and plots, as an action either remains in a role-specific plot

or not. In the former case an actor in this role is in principle eligible for executing the corresponding action; in the latter case he or she is not.

Let us now look briefly at a more sophisticated way to express rights and obligations that depend on other actions. An *obligation* specifies what an actor in a particular role has to do. A *right* specifies what an actor in a particular role is permitted to do. Both obligations and rights together lead to complex deontic integrity constraints. We use the following logical language \mathcal{L} for this purpose:

- All propositional atoms are also atoms of \mathcal{L} .
- If α is an action on scene s and r is a role associated with s , then $\mathbf{O} \text{ } do(r, \alpha)$ is an atom of \mathcal{L} .
- If α is an action on scene s and r is a role associated with s , then $\mathbf{P} \text{ } do(r, \alpha)$ is an atom of \mathcal{L} .
- If α is an action on scene s and r is a role associated with s , then $\mathbf{F} \text{ } do(r, \alpha)$ is an atom of \mathcal{L} .
- For $\varphi, \psi \in \mathcal{L}$ we also have $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \rightarrow \psi$ and $\varphi \leftrightarrow \psi$ are also formulae in \mathcal{L} .

The interpretation is standard. In particular, $\mathbf{O} \text{ } do(r, \alpha)$ means that an actor with role r is obliged to perform action α , $\mathbf{P} \text{ } do(r, \alpha)$ means that an actor with role r is permitted to perform action α , and $\mathbf{F} \text{ } do(r, \alpha)$ means that an actor with role r is forbidden to perform action α .

Example 3.22. The on-line loan example plot from Example 3.15 only contains one role customer, thus all actions and scenes in this example are associated with this role. Nevertheless, we may express some deontic constraints using the logical language defined above.

A customer has the obligation to leave his or her details (action α_8), once a personal loan or a mortgage has been selected (condition φ_6 or φ_7). Furthermore, if a mortgage is selected (condition φ_7), the customer is obliged to describe securities (action α_{16}) and to enter obligations (action α_{17}), i.e., we obtain the deontic constraints

$$\varphi_6 \vee \varphi_7 \rightarrow \mathbf{O} \text{ } do(\text{customer}, \alpha_8)$$

and

$$\varphi_7 \rightarrow \mathbf{O} \text{ } do(\text{customer}, \alpha_{16}) \wedge \mathbf{O} \text{ } do(\text{customer}, \alpha_{17})$$

Furthermore, a customer is allowed to look at mortgage samples (action α_6), which is simply expressed by $\mathbf{P} \text{ } do(\text{customer}, \alpha_6)$.

Let us now extend the plot by bringing in operations that are associated with a role clerk. If p_1 denotes the plot in Example 3.15, we can now look at the plot $p = p_1 \square p_2$, in which the process p_2 is defined as follows:

```
( enter_loan_assessment ;
  ( (select_pl_application ; assess_pl_application { $\varphi_{13} \vee \varphi_{14}$ } ;
    ( { $\varphi_{13}$ } ( ( reject_pl_application { $\varphi_{15}$ } )  $\square$ 
      ( ( { $\neg\varphi_{14} \wedge \neg\varphi_{15}$ } explore_pl_alternative ;
        ( modify_pl_application  $\square$  skip ))* { $\varphi_{14} \vee \varphi_{15}$ } )))  $\square$ 
    ( { $\varphi_{14}$ } ( ( accept_pl_application { $\varphi_{16}$ } )  $\square$ 
      ( explore_pl_alternative ;
        ( modify_pl_application  $\square$  skip ))* { $\varphi_{16}$ } ))))  $\square$ 
  ( select_m_application ; assess_m_application { $\varphi_{17} \vee \varphi_{18}$ } ;
    ( ( { $\varphi_{17}$ } arrange_meeting { $\varphi_{19} \vee \varphi_{20}$ } )  $\square$ 
      ( { $\varphi_{18}$ } accept_m_application { $\varphi_{20}$ } )))))

```

This involves the additional conditions

$$\begin{array}{ll} \varphi_{13} \equiv \text{pl_critical} & \varphi_{14} \equiv \text{pl_acceptable} \\ \varphi_{15} \equiv \text{pl_rejected} & \varphi_{16} \equiv \text{pl_accepted} \\ \varphi_{17} \equiv \text{m_critical} & \varphi_{18} \equiv \text{m_acceptable} \\ \varphi_{19} \equiv \text{mortgage_rejected} & \varphi_{20} \equiv \text{mortgage_accepted} \end{array}$$

We use the following abbreviations for the new actions

$$\begin{array}{ll} \alpha_{22} = \text{enter_loan_assessment} & \alpha_{23} = \text{select_pl_application} \\ \alpha_{24} = \text{assess_pl_application} & \alpha_{25} = \text{reject_pl_application} \\ \alpha_{26} = \text{explore_pl_alternative} & \alpha_{27} = \text{modify_pl_application} \\ \alpha_{28} = \text{accept_pl_application} & \alpha_{29} = \text{select_m_application} \\ \alpha_{30} = \text{assess_m_application} & \alpha_{31} = \text{arrange_meeting} \\ \alpha_{32} = \text{accept_m_application} & \end{array}$$

As these new actions are associated with the role clerk, the plot p_2 is a personalisation of the plot p for this role. Analogously, p_1 is a personalisation of the plot p for the role customer. Then

$$\varphi_{10} \vee \varphi_{11} \rightarrow \mathbf{O} \text{ do}(\text{clerk}, \alpha_{22})$$

specifies that if a personal loan or mortgage has been applied for, a bank clerk will have to enter the loan assessment part of the system. This links customers to bank clerks, in particular, as φ_{10} corresponds to a completed application for a personal loan by a customer, and φ_{11} corresponds to a completed application for a mortgage. If these were simple actions, say α and β , respectively, they could have been expressed by $\varphi_{10} \leftrightarrow \text{do}(\text{customer}, \alpha)$ and $\varphi_{11} \leftrightarrow \text{do}(\text{customer}, \beta)$, but we only permitted simple actions in the deontic atoms.

$$\varphi_{13} \rightarrow \mathbf{P} \text{ do}(\text{clerk}, \alpha_{25}) \wedge \mathbf{P} \text{ do}(\text{clerk}, \alpha_{27})$$

expresses that if the assessment of a personal loan shows that the application is critical, the bank clerk is permitted to reject the application. He or she is

also permitted to modify the application. According to the story space, only one of these activities will be taken. Similarly,

$$\varphi_{14} \rightarrow \mathbf{F} \text{ do(clerk, } \alpha_{25}) \wedge \mathbf{P} \text{ do(clerk, } \alpha_{27})$$

specifies that if the assessment of a personal loan shows that the application is acceptable, the bank clerk is not permitted to reject the application. However, he or she may still modify the application, e.g., if this turns out to be advantageous for the customer.

$$\varphi_{18} \rightarrow \text{do(clerk, } \alpha_{32}) \quad \text{and} \quad \varphi_{17} \rightarrow \text{do(clerk, } \alpha_{31})$$

express that in case a mortgage application is acceptable, the bank clerk will accept it, whereas in case it turns out to be critical the bank clerk will arrange a meeting to discuss the problems and possible options.

$$\text{do(clerk, } \alpha_{31}) \wedge \neg \text{do(customer, } \alpha_1) \rightarrow \varphi_{19}$$

specifies that if a bank clerk arranges a meeting to discuss mortgage options, but the customer does not enter the loan system for a revised application, the mortgage application will be rejected.

We may of course extend the on-line loan system further by adding a role `mortgage_advisor` and further deontic constraints.

The *goal* of an actor can be expressed by a postcondition to the story space. We may then ask how to generalise personalisation to goals of actors. We expect to discard those parts of the plot that do not contribute to reaching the goal, i.e., a postcondition ψ . We will discuss a formal approach to computing such a goal-specific personalised plot in Chapter 4.

Example 3.23. For the Example 3.15 we may think of an actor who is going to apply for a home loan. This can be expressed by the goal φ_{10} .

3.2.3 User Profiles and Types

While roles classify actors according to their rights and obligations, *user profiles* and *user types* classify actors according to their behaviour. That is, roles indicate a proactive approach to WIS modelling, whereas user profiling is reactive.

The general approach is to start with characterising properties of users and to provide values for each of these properties. Each combination of such values defines a user profile. However, the behaviour for some of these profiles is usually the same. So we combine user profiles to user types.

Furthermore, each user profile or type leads to rules, in particular preference rules. According to the characterising aspects of WISs we may distinguish between preferences concerning the content, the functionality or the presentation. In the context of storyboarding such preferences can be expressed by constraints on the story space. At lower levels of abstraction we can add details to these constraints or add constraints that cannot yet be expressed on a high level of abstraction.

Dimensions of User Profiles

We will now discuss in more detail these user profiles and types. We start with a finite set Δ of *dimensions* capturing the properties of users. For each dimension $\delta \in \Delta$ we assume to be given a *domain* $dom(\delta)$. Some of these domains may be totally ordered, i.e., there is a total order \leq_δ on $dom(\delta)$ – usually, we drop the index. A totally ordered domain is also called a *scale* of the dimension.

Example 3.24. Take $\Delta = \{\text{experience}, \text{skill}, \text{training}, \text{presentation_preferences}, \text{goal_orientation}\}$ as the set of dimensions. The dimension experience describes how experienced a user is with respect to the application. For instance, in loan applications users might have applied for loans or mortgages several times and could therefore be counted as being experienced, whereas others might be novices. This would lead to a scale for experience

$$dom(\text{experience}) = \{\text{novice}, \text{average}, \text{experienced}\}$$

with the order $\leq_{\text{experience}}$ defined by

$$\text{novice} \leq \text{average} \leq \text{experienced}.$$

The dimension goal-orientation refers to how much focused a user is with respect to a particular, i.e., whether he or she just looks around for an opportunity, explores side tracks or approaches directly the goal, e.g., applying for a mortgage. This could be modelled by a scale

$$dom(\text{goal-orientation}) = \{\text{surfer}, \text{navigator}, \text{searcher}\}$$

with $\text{surfer} \leq \text{navigator} \leq \text{searcher}$. The dimension presentation-preferences can be used to distinguish between users who prefer to see all details and those who first want to see a condensed summary. In this case we can use the scale

$$dom(\text{presentation_preferences}) = \{\text{detailed}, \text{normal}, \text{condensed}, \text{terse}\}$$

with $\text{detailed} \leq \text{normal} \leq \text{condensed} \leq \text{terse}$.

We may also use the dimensions to indicate the location of a user, if this impacts on the WIS. For instance, in choosing payment options it is common for American users to use cheques, whereas Europeans might prefer a direct debit. Thus, we have a dimension location or simply country with an obvious domain, i.e., the set of all countries. This gives an example of a dimension with a non-ordered domain.

Using location as a dimension is related to the original question, where the WIS will be used. In the same way we may use a dimension capturing a time-scale, which addresses the question, when the WIS will be used.

Identifying Dimensions

The dimensions used in user profiles depend both on the individual and the application. *Social demographic characteristics* of users such as age, gender, social status, education, profession, cultural and ethnical binding, religious binding, moral and ethical values, etc. can give rise to dimensions, if it is expected that they impact on the usage of the system. That is, a characteristic is only used as a dimension for user profiling, if differences with respect to this characteristic imply preferences concerning the content, the functionality or the presentation.

This applies in the same way to general *personality characteristics* such as open-mindedness, emotionality, objectiveness, patience, willingness to take risks, etc.

Application-dependent dimensions can be identified from the following sources:

- the ability to search for solutions, solve problems, detect and resolve conflicts, schedule work tasks;
- the communication skills and computer literacy;
- the knowledge and education level regarding the task domain;
- the frequency and intensity of system usage;
- the way information is handled, i.e., the direction of the information flow, the necessary and optional input, the intended information usage, the amount and size of information and the complexity of information;
- the experience in working with the system and with associated tasks.

In the area of e-learning the profiling of users, i.e., the identification of learner types, has been investigated more thoroughly. For instance, [742] identifies characteristics for the learning style, the learners, the attitude towards learning tasks and the examination style. The following is a non-exhaustive list of these characteristics that can be used to define dimensions:

- For the learning style we have the following general characteristics:
 - Guidance: The degree to which the learner prefers being guided or not in the learning process.
 - Visual Modality: The degree to which the learner prefers a presentation in visual manner.
 - Auditory Modality: The degree to which the learner prefers a presentation in auditory manner.
 - Textual Code: The degree to which learners with visual learning style prefer having access to text.
 - Illustrational Code: The degree to which learners with visual learning style prefer having access to images.
 - Example: The degree to which the learner prefers learning deductively or inductively.
- Characteristics that refer to the learners are:

Persistency: The degree to which the learner in general can or cannot mentally cope with new material.

Retentivity: The degree to which the learner in general can or cannot memorize learned material.

Computer literacy: The degree to which the learner has or has not acquired skills in using modern computing infrastructure for a task at hand.

Curiosity: The degree to which the learner in general is or is not fond of learning new material.

- Learning task related characteristics are the following:

Prerequisites: The degree to which the learner has or has not learnt the required subjects.

Performance: The degree to which the learner has or has not performed well regarding background subjects, required subjects or subjects in general.

Motivation: The degree to which the learner is or is not interested in efficiently mastering the learning task at hand.

Confidence: The degree to which the learner believes to be capable or incapable of successfully mastering the learning task.

- Characteristics that apply to the examination style are:

Kind: The degree to which the learner either prefers his knowledge or his skills or a combination of both being checked.

Control: The degree to which the learner prefers his increment in knowledge or skills being checked.

Evaluation strategy: The degree to which the learner prefers his solution to an examination problem being evaluated right after its submission or whether he or she prefers all problem solutions being evaluated at the same time.

Feedback: The degree to which the learner prefers receiving full explanations of the assessments of his or her problem solutions or whether he or she prefers being notified of the correct results only.

Building User Types

As there are usually many dimensions, each of which has a domain with at least two elements. This gives us many ways to combine these values to obtain user profiles. In order to be able to manage the combinatorial explosion, we introduce user types.

Definition 3.25. Let $\Delta = \{\delta_1, \dots, \delta_n\}$ be a set of dimensions. Then the *set of user profiles* (or the *user-grid*) over Δ is $gr(\Delta) = dom(\delta_1) \times \dots \times dom(\delta_n)$. A *user type* over Δ is a subset $U \subseteq gr(\Delta)$.

This way of defining a user type as any subset of the user-grid leaves a lot of freedom to define a set $\{U_1, \dots, U_k\}$ of user-types for a particular story space.

We must of course ensure that this set of user types is *complete* in the sense that all user profiles are covered. That is, for each user profile $(v_1, \dots, v_n) \in gr(\Delta)$, i.e., $v_i \in dom(\delta_i)$, there must exist at least one $j \in \{1, \dots, k\}$ with $(v_1, \dots, v_n) \in U_j$.

One might expect that user profiles that contribute to the same user type, do not differ too much in the values for the dimensions. For this consider cubes in the user-grid.

Definition 3.26. Let $\Delta = \{\delta_1, \dots, \delta_n\}$ be a set of dimensions. A subset $U \subseteq gr(\Delta)$ is a *cube* iff it has the form $U = D_1 \times \dots \times D_n$, such that for all $i = 1, \dots, n$, for which $dom(\delta_i)$ is totally ordered scale, whenever $v_{i1}, v_{i2} \in D_i$ with $v_{i1} \leq_{\delta_i} v_{i2}$ holds, then also $v_i \in D_i$ for $v_{i1} \leq_{\delta_i} v_i \leq_{\delta_i} v_{i2}$.

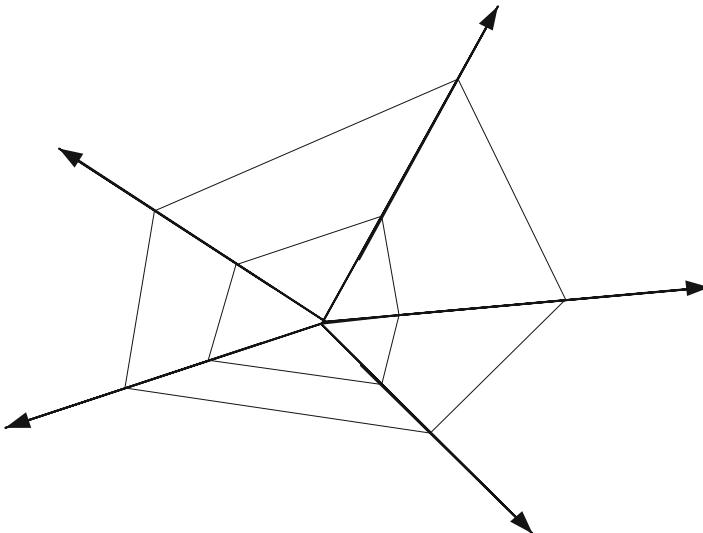


Fig. 3.12. Kiviat graph of user types

Thus, one common way of defining user types is to consider cubes in the user-grid. We call these user types *kiviat*, because they can be easily represented by *Kiviat graphs* as shown in Figure 3.12.

Another common way exploits aggregate functions $f : gr(\Delta) \rightarrow \mathbb{N}$ and intervals $I_i = [a_i, b_i] \subseteq \mathbb{N}$ or $I_i = [a_i, \infty)$. These can be used to define *aggregate user types* of the form

$$U_i = \{(v_1, \dots, v_n) \in gr(\Delta) \mid f(v_1, \dots, v_n) \in I_i\}.$$

We already added user types to the story space by assigning a set of user types to each scene. This indicates which stories will be supported for

which user profiles. We may then build user-type-specific scenarios and plots, respectively, in the same way as we build role-specific scenarios and plots.

Preference Rules

The major purpose of introducing user types is to provide ways to customise the WIS to its users, i.e., to personalise the system. In principle there are two ways of doing it. The first one would just introduce user-type-specific plots, whereas the second one would try to generate them out of preference rules and general rules about the story space. Therefore, it is a good idea to formulate preference rules. Even if user-specific plots are not derived but specified by WIS designers, the rules can be useful to quality check the customised plots.

In view of the algebraic constructors, that were used to specify plots, we consider preference rules in connection with these constructors. This leads to the following definition.

Definition 3.27. Let U be a user type. A *preference rule* associated with U is expressed through one of the following equations:

1. An equation of the form $\{\varphi\}(\alpha \square \beta) = \{\varphi\}\alpha$ expresses that a user conditionally prefers action α over action β , where the condition is expressed by φ . The special case $\varphi = \text{true}$ expresses an unconditional preference.
2. An equation $p; (p_1 \square p_2) = p; p_1$ expresses the conditional preference of the process p_1 over p_2 after the process p . The special case $p = \text{skip}$ expresses an unconditional preference.
3. An equation $p_1; p_2 \square p_2; p_1 = p_1; p_2$ expresses a preference of order, i.e., if the processes p_1 and p_2 can be executed in arbitrary order, it is preferred to execute first p_1 .
4. An equation $p^* = p; p^*$ expresses that in case of an iteration it will be executed at least once.

If a storyboard description catalogue is used, this can now be complemented by user types in the following way:

User Type: ⟨name of the user type⟩

Definition: ⟨list of dimension specifications or aggregate definition⟩

Preferences: ⟨list of preference rules⟩

Here, each dimension specification has either the form

dimension ⟨dimension name⟩ between ⟨value⟩ and ⟨value⟩

or

dimension ⟨dimension name⟩ in ⟨list of values⟩,

while an aggregate definition takes the form

⟨function name⟩(⟨dimension list⟩) between ⟨value⟩ and ⟨value⟩ .

So the personalisation problem with respect to the preference rules of a user type is to find a simpler plot by applying these equations. In Chapter 4 we will look at a formal algebraic approach to solve this problem. The techniques will allow the personalisation with respect to preferences to be combined with the personalisation with respect to goals.

In addition to the preference rules that are associated with a user type, we obtain also general rules – sometimes called “application knowledge” that describe equational dependencies among actions and conditions in a plot. Examples of such rules are the following:

- An equation of the form $\{\neg\varphi\}\alpha = \text{fail}$ (or equivalently $\alpha = \{\varphi\}\alpha$) expresses that the condition φ is a precondition for the action α .
- An equation of the form $\alpha\{\neg\varphi\} = \text{fail}$ (or equivalently $\alpha = \alpha\{\varphi\}$) expresses that the condition φ is a postcondition for the action α .
- An equation of the form $\alpha\{\varphi\} = \{\varphi\}\alpha$, which is equivalent to $\{\neg\varphi\}\alpha = \alpha\{\neg\varphi\}$ and to $\{\varphi\}\alpha\{\neg\varphi\}\square\{\neg\varphi\}\alpha\{\varphi\} = \text{fail}$, expresses that the condition φ (and so its negation $\neg\varphi$) is invariant under the action α .
- An equation of the form $\varphi \wedge \psi = \text{false}$ expresses that the conditions φ and ψ exclude each other.

Here we used an additional process *fail*, which is never defined. The claimed equivalences will be discussed in Chapter 4.

Example 3.28. Let us first look at the simple plot in Example 3.14. Assume we use only a single dimension location with

$$\text{dom}(\text{location}) = \{\text{America}, \text{Europe}, \text{Asia}\}.$$

This gives us three obvious user types American, European, and Asian. For an American the bank transfer option should be invalid, which gives the equation $\alpha_3 = \text{fail}$. Note that this is a postcondition equation with the postcondition *false*.

For a European the option to pay by cheque should be invalidated. So we obtain the equation $\alpha_4 = \text{fail}$. Furthermore, assume that Europeans prefer to pay by credit card, if this turns out to be a valid option. For this we use the equation

$$\{\varphi_1\}(\alpha_3 \square \alpha_4) = \text{fail}.$$

This example shows that the definition of preference rules usually depends only on a few dimensions. For instance, values of dimensions other than location are irrelevant for the choice of the payment option. Hence, we could have taken the values of the dimension as part of the rules, which would give us conditional rules such as

$$\begin{aligned} \text{location} = \text{America} &\rightarrow \alpha_3 = \text{fail} \\ \text{location} = \text{Europe} &\rightarrow \alpha_4 = \text{fail} \end{aligned}$$

and

$$\text{location} = \text{Europe} \rightarrow \{\varphi_1\}(\alpha_3 \square \alpha_4) = \text{fail}$$

We will discuss implications of using such conditional rules in Chapter 4 in the context of personalisation by term rewriting.

3.3 Task Modelling

The story space describes the usage of a WIS on a high level of abstraction. It specifies what is used and how it is used. The actors specify who uses the WIS, and to a minor extend also, where and when the WIS is used. Tasks are used to model what the usage is for, i.e., why the WIS should be used. We now address this last part of storyboards, the tasks. Tasks can be performed by a single user or they can be the cooperative effort of several users. That is, we consider WISs to be task-oriented systems.

Tasks describe the general purposes of the WIS. They have already been identified on the strategic level, which we discussed in Chapter 2. In particular, each task is associated with a goal. Tasks combine roles that are involved in the task, actions executed by actors in these roles, consequently scenes to which these actions belong, information consumed by the actions, and data flowing between the actions. In addition, there may be a precondition and an event that triggers the task. Actions can be grouped together into subtasks to provide a more concise form of task specification. Thus, a task is largely specified by what has already been defined for the story space including the integrity constraints that define rights, obligations, intentions and behaviour.

For instance, a task in a home loan system can be to submit a mortgage application. This involves only one actor in a single role customer. In an extended system, the task may be to submit, approve and implement a mortgage, in which case other actors in the role of a bank clerk or a mortgage advisor would participate in the task.

3.3.1 Tasks and Subtasks

Let us first define the notion of task formally. In Section 3.3.2 we will then again look at ways to represent tasks.

Definition 3.29. A *task* τ in a story space consists of

- a set $act(\tau) = \{\tau_1, \dots, \tau_n\}$ of subtasks, which are tasks or actions on the story space,
- a goal $\gamma(\tau)$, which is a postcondition to the story space,
- a precondition, which is another Boolean condition φ on the story space,
- and a triggering event $ev(\tau)$ of the form $do(r, \alpha)$ with a role r and an action α .

Furthermore, with each subtask τ_i we associate a set of scenes and a set of roles. If τ_i is atomic, i.e., an action α , then it will be associated with exactly one scene s and exactly one role r . In this case, we also associate a view $V_i = (\mathcal{S}_i, q_i)$ with $\tau_i = \alpha$, which is a view over the view V_s . Furthermore, with any pair of subtasks (τ_i, τ_j) ($i \neq j$) we associate another view $V_{ij} = (\mathcal{S}_{ij}, q_{ij})$ describing the data communicated from subtask τ_i to subtask τ_j .

Example 3.30. Look again at the loan application system from Example 3.15. In this system we may have a task $\tau = \text{application_for_mortgage}$ with precondition $\mathcal{P}\text{re}(\tau) = \text{true}$ and triggering event $\mathcal{E}\text{v}(\tau) = \text{do}(\text{customer}, \alpha_1)$, i.e., the task is triggered when a customer enters the loan system. The goal of the task is the condition $\gamma(\tau) = \varphi_{11}$, i.e., the customer should apply for a mortgage.

In this case the set of subtasks is simply a set of actions. We have

$$\mathit{act}(\tau) = \{\alpha_1, \alpha_2, \alpha_4, \alpha_6, \alpha_8, \alpha_{11}, \alpha_{13}, \alpha_{14}, \alpha_{15}, \alpha_{16}, \alpha_{17}, \alpha_{18}, \alpha_{19}, \alpha_{20}, \alpha_{21}\},$$

i.e., it is the set of actions that would also arise from a personalisation with the goal φ_{11} . We will see this relationship in Chapter 4.

The scenes are as specified in Example 3.15, and the associated role for all subtasks is always customer.

Of course, the deontic constraints specified for the story space apply to each task specification.

3.3.2 Representation Means for Tasks

As already remarked, a task is already largely specified by what was already defined for the story space. In particular, a task defines a set of actions associated with it. These are first all the actions in $\mathit{act}(\tau)$ and all actions that belong to subtasks. As all these actions belong to some scene, a task is also associated with a set of scenes. Taking both together each task τ defines a scenario $\mathcal{E}(\tau) = (\mathcal{S}, \mathcal{A}, \sigma, \tau)$.

Example 3.31. The scenario for the task `application_for_mortgage` in Example 3.30 is shown in Figure 3.13. This scenario defines a subgraph of the one in Figure 3.6.

In the same way we may define plots for tasks. We may then exploit all alternative representation means such as communication matrices, statecharts and the storyboard catalogue for the representation of tasks. For instance, in the storyboard catalogue a task would be represented as follows:

Task:	$\langle \text{name of the task} \rangle$
Goal:	$\langle \text{condition} \rangle$
Subtasks:	$\langle \text{list of subtask names} \rangle$
Actions:	$\langle \text{list of action names} \rangle$
Precondition:	$\langle \text{condition} \rangle$
Event:	$\langle \text{action} \rangle \text{ by } \langle \text{role name} \rangle$
Scenario:	$\langle \text{scenario name} \rangle$

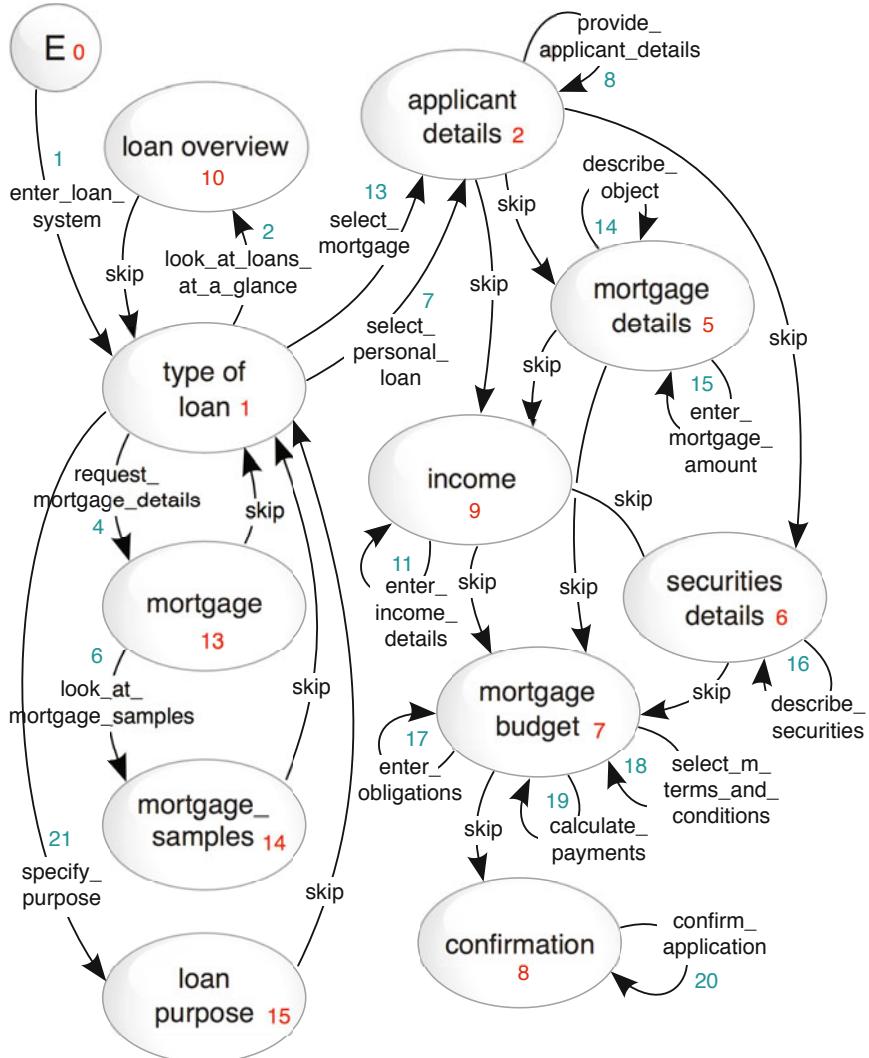


Fig. 3.13. Scenario for the task `application_for_mortgage`

As tasks are composed of subtasks, and the “elementary” subtasks are actions, we can use trees to represent this relationship. We obtain a *task decomposition tree*.

We may, however, add other relationships between tasks. For instance, the task `application_for_mortgage` in Example 3.30 is a specialisation of the task `application_for_loan`. That is, we obtain a relationship \sqsubseteq on tasks expressing *specialisation*. Furthermore, the completion of a task may be necessary to

enable another task to be started. That is, we obtain another relationship \rightsquigarrow on tasks expressing *enabling*.

With respect to the storyboard catalogue we may extend the description of tasks as follows:

Task:	\langle name of the task \rangle
...	
Enables:	\langle list of task names \rangle
Specialises:	\langle list of task names \rangle

3.4 The Complete View of Storyboards

Let us now summarise this chapter on storyboarding. A storyboard is a high-level specification of the usage of a WIS. It consists of a story space, a set of actors, and a set of tasks. In view of the specifications in this chapter we finally obtain the following definition.

Definition 3.32. A *storyboard* consists of

- a set of scenarios, one of which is the main scenario, whereas the others define scenes;
- a plot specified by a SiteLang process;
- a set of roles;
- a set of user types;
- a set of tasks each associated with a goal;
- and a set of constraints comprising deontic constraints for the rights and obligations of roles, preference rules for user types, and other dependencies on the plot.

Each *scenario* provides an application-oriented view of parts of the storyboard. In particular, the main scenario provides an application-oriented view of the whole storyboard. Some of the scenarios describe particular tasks. A *scene* provides a location-oriented view of parts of the storyboard. The *plot* provides an action-oriented view of the whole storyboard. Parts of the plot define an action-oriented view of a scene or task.

For each *role* or *user-type* there are role-specific or user-type-specific scenarios and plots, respectively. These provide actor-oriented views of the storyboard. Finally, each *goal* defines a goal-specific plot, which gives a goal-oriented view of the storyboard.

The constraints on the storyboard are expressed on a propositional level. Constraints govern the relationships between the various parts of the storyboard.

Given a storyboard specification, the critical question is, whether it makes sense, i.e., whether it really captures the usage of the WIS on a high-level of abstraction. With respect to the strategic WIS model the following quality questions arise:

- Is it possible to achieve the goals associated with the tasks of the WIS?
- Is it possible for a user to achieve his or her goals?
- How does the specification of the storyboard customised to a particular role, user type or goal look like? Is this in accordance with what we would expect from the behaviour of a corresponding user?

In addition, the storyboard specifies on a rudimentary, sketchy level, how the scenes in the story spaces have to be supported. This specification of data consumption, production and communication data associated with actions together whether the actions, their pre- and postconditions and their enabling and triggering events form the basis of the specification of media types as the supporting means for scenes.

3.5 Bibliographical Remarks

Storyboarding in a process-oriented holistic manner focusses on user intentions. The conceptual model of storyboarding (see e.g., [727]) takes this up by providing an integrated model comprising the story space capturing the stories and the plot, actors, and tasks. The fist attempts to storyboard [203, 850] have been based on our experience obtained during development of city, region and other infotainment systems, see also [844, 501, 226, 479, 722, 736]. It has been extended in order to cope with edutainment, e-government, community and e-business system, e.g., [177, 87, 226, 371, 429, 396, 709, 742, 687, 609, 796, 878]. The formal specification discussed in this chapter has been applied to these projects. The SiteLang language is a result of our investigations. The approach can be extended to services in general [27, 77, 174, 533, 532, 748, 750]. Communication matrices are widely applied in the industrial practice (see also [335, 439, 702]).

The first variant of the storyboard editor is sketched in [858]. Short surveys are provided in [598, 573].

Inspired by approaches in theatre and film [228, 887, 901], the story space comprises scenes and the actions in these scenes, and the plot describes the details of the action scheme [49, 875, 879]. Furthermore, the model describes the actors in these scenes, i.e., groups of users, which leads to roles, profiles, goals, preferences, obligations, and rights [866]. The actors are linked to the story space by means of tasks [132, 133].

Some of the approaches (see, e.g., [47, 55, 106, 269, 260, 311, 686, 762]) miss out on the important aspect of storyboarding, which is needed to capture the business content of the system. Instead business process modelling on the basis of BPMN [912] or activity diagrams are used. However, technical systems are different from social systems. Therefore, their modelling should not be mixed with the other side. Social systems must be far more flexible. Dynamic BPMN [666] has been introduced to overcome this limitation. A more flexible approach is generic processes [317, 872].

User modelling is an essential element of website development [400, 730, 936]. User models can be developed based on three specific profiles: *education*, *work* and *personality* profiles [360]. Actor models generalise the notion of actors used in [8]. A formal description of interactive systems based on user profiling has been introduced in [30, 802]. A *portfolio* of a user combines responsibilities and a collection of tasks assigned to or intended by a user and for which he or she has the authority and control, and a description of involvement within the task solution [730, 731, 740]. The concept of the information portfolio stems from information logistics research [188, 297, 905] and has got nowadays a wide acknowledgement, e.g., [624]. Manipulation requests must explicitly be specified since views might be not updateable (see information system course books, e.g., [46]). Modularisation is a common concept in computer science. One of its elements are hierachic or incremental construction. Task models are an essential element in participatory design (e.g., [635]) and in human-computer interface description (e.g., [258, 602, 634, 867]).

WIS development uses extreme programming ideas for early testing of systems with potential users [604] and user group adaptation and sophistication [661, 676, 874]. User modelling may also start with user observation (e.g., [913]) or user behaviour mining (e.g., [800, 931]). The storyboarding approach can be combined with automatic derivation and adaptation of user interfaces, e.g., [295, 933]. The distinction between goals and soft goals is introduced in the i* approach, e.g., [275].

Storyboarding become nowadays an acknowledged technique [141, 293, 296, 374, 466, 474, 892, 622, 813, 863] although specific languages or tools are not yet developed except [331, 608, 791, 858, 938]. The narrative dimension became the kernel of storyboards [49, 52, 152, 288, 696, 812, 875, 879].

Users are limited in their attention and their memory [285, 570, 699, 760, 801]. This restriction is taken into account in the user model we have introduced. User models are based on cognitive psychology (e.g., [606]), personalisation techniques (e.g., [35, 652, 725, 734]), usability engineering (e.g., [626, 797]), context adaptation techniques (e.g., [12, 89, 114, 294, 395, 601, 605, 860]), communication theory (e.g., [24, 145, 310, 316, 712, 713], and the corresponding life cases [732, 735]. User models may also use stereotypes or pattern [474]. Life cases have already been envisioned in [140] and formalise scenarios used in software engineering and user modelling [163].

Key Messages

A **storyboard** comprises a *story space*, which

- models the *usage* of a web information system by specifying possible navigation paths between scenes;

- captures the *user activities* by actions associated with the navigation paths;
- captures enabling and constraining *conditions* for the possible navigation paths;
- defines the *plot* (or *action scheme*) of the web information system.

A **Storyboard** further models its *actors* comprising

- a classification by *user types* characterised by profiles and associated with *preferences*;
- a classification by *roles* associated with *rights and obligations* that are expressed by deontic constraints.

A **Storyboard** is complemented by *tasks* that specify for which purposes the web information system is to be used.



Semantics and Inferences on Storyboarding ♣

In this chapter we take a closer look at formal aspects of storyboarding. This is considered as advanced material, which readers who are primarily interested in the development methodology may skip.

In Section 4.1 we round up the presentation of storyboarding by a deeper formal treatment with applications to story space personalisation, and in Section 4.2 we relate personalisation to the deontic constraints associated with roles.

Customisation to preferences and personalisation started with [722] and was further developed in [725, 728, 737] with emphasis on application categories in [744, 714]. The work in [734] summarises the formal customisation approach with formal foundations being further investigated in [745]. Reasoning about deontic action constraints was initially handled in [743, 727].

4.1 Story Algebra and Personalisation

So far we first defined the story space as a hierarchically organised collection of scenarios. As there is one main scenario, while other scenarios specify scenes in more details, this collection is actually a tree. Using the substitution of scenarios for the scenes they specify, we can flatten this hierarchy to a single scenario, which only contains elementary scenes. As such a scenario is basically a directed graph, we consider the paths from the start scene to some final scene. These paths define sequences of user actions, i.e., they are actually the stories we want to model. In this way we define the base semantics of a storyboard.

This base semantics is first refined by the various constraints of the story space that are defined by pre- and postconditions, acceptance conditions and events, and enabling and triggering events. These constraints are global and do not depend on actors. They can be captured by the plot, which drastically reduces the set of valid stories. However, as the plot is defined independently using `SiteLang`, we have to ensure that these constraints are indeed subsumed

by the plot specification. We can always ensure that the stories enabled by a plot are compatible with the scene transition relation τ of the corresponding scenario. We can also ensure that pre- and postconditions and acceptance conditions are captured by the plot, as this simply requires the addition of Boolean conditions. Likewise, enabling and triggering events and acceptance events may require the addition of actions to the plot, thus can also be captured by it. This implies that any further investigation of semantics should be based on the plot.

Furthermore, from the specification of actors we obtain roles and user types. The latter ones are associated with preference rules, which further restrict the plot, though this time in dependence of the user type. The customisation of the plot according to these preferences gives a precise meaning to *personalisation* with respect to functionality on the level of storyboarding. This should subsume user types associated with scenes, and can be extended by the actual goal of a user.

We now deal with these aspects of the semantics of storyboarding. We first formalise the set of stories enabled by a plot and then address the problem of personalisation with respect to preference rules and goals. This leads us to inferences by term rewriting on Kleene algebras with tests, for which we examine desirable properties such as termination and Church Rosser property.

A second aspect of storyboard semantics arises from the rights and obligations that are associated with roles. We introduced a propositional deontic logic to formalise these constraints. Next we formalise the semantics of this logic by means of feasible status sets and extend the term rewriting approach on the plot to decide whether the preference rules of an actor are compatible with the deontic constraints.

4.1.1 Formalisation of Scenarios and Plots

Concentrating on the plot we first investigate the mathematics behind story algebra, which will lead us to *Kleene algebras with tests* [452]. If we use $+$ to denote a choice (\square in `SiteLang`), \cdot to denote a sequence ($;$ in `SiteLang`) and $*$ to denote iteration, we have to look at a mathematical structure with two binary and one unary operator. In addition we use 1 for an action that does nothing (called `skip` in `SiteLang`), and 0 for an undefined action (usually called `fail` or `abort`). Then it is not difficult to see that these operators satisfy the axioms of a Kleene algebra.

Definition 4.1. A *Kleene algebra* (KA) \mathcal{K} [421] consists of a carrier-set K containing at least two different elements 0 and 1 , a unary operation $*$, and two binary operations $+$ and \cdot such that the following axioms are satisfied:

- $+$ and \cdot are associative, i.e., for all $p, q, r \in K$ we must have $p + (q + r) = (p + q) + r$ and $p(qr) = (pq)r$;
- $+$ is commutative and idempotent with 0 as neutral element, i.e., for all $p, q \in K$ we must have $p + q = q + p$, $p + p = p$ and $p + 0 = p$;

- 1 is a neutral element for \cdot , i.e., for all $p \in K$ we must have $p1 = 1p = p$;
- for all $p \in K$ we have $p0 = 0p = 0$;
- \cdot is distributive over $+$, i.e., for all $p, q, r \in K$ we must have $p(q + r) = pq + pr$ and $(p + q)r = pr + qr$;
- p^*q is the least solution x of $q + px \leq x$ and qp^* is the least solution of $q + xp \leq x$, using the partial order $x \leq y \equiv x + y = y$.

We adopted the convention to write pq for $p \cdot q$, and to assume that \cdot binds stronger than $+$, which allows us to dispense with some parentheses. We will write $\mathcal{K} = (K, +, \cdot, ^*, 0, 1)$ to denote a Kleene algebra. Elements of K will be called *processes*.

KAs originate from formal language theory; they were introduced by Kleene as an algebraic theory for studying finite automata and regular languages [421]. Of course, the standard example is regular sets. For other non-standard examples refer to [449] and [451]. However, as processes share the same algebraic properties for sequencing, choice, interleaving and iteration [241, 242], expressions in KAs can be interpreted by assignment-free processes.

Here, we want to use Kleene algebras to represent story algebras as discussed in the previous chapter. We can represent parallelism by equating $p\|q$ with $pq + qp$, so in fact we are interested in interleaved, concurrent processes, when we talk of parallel execution. In particular, as semantics is defined by the valid stories that are permitted by a plot, this is no loss of generality. However, we still need an extension to capture guards and post-guards, and we have to handle associated scenes. Capturing guards and post-guards leads to Kleene algebras with tests, which were introduced in [452].

Definition 4.2. A *Kleene algebra with tests* (KAT) \mathcal{K} [452] consists of a Kleene algebra $(K, +, \cdot, ^*, 0, 1)$, a subset $B \subseteq K$ that contains 0 and 1 and is closed under $+$ and \cdot , and a unary operation $\bar{}$ on B , such that $(B, +, \cdot, \bar{}, 0, 1)$ forms a Boolean algebra.

We write $\mathcal{K} = (K, B, +, \cdot, ^*, \bar{}, 0, 1)$. Of course the elements of B are the “tests”, and the operator $\bar{}$ expresses negation. Using the theory of KATs for personalisation is of particular interest, as it subsumes the theory of propositional Hoare logic [321] as shown in [453].

Note that the operator $+$ applied to two tests $\varphi, \psi \in B$ represents the logical OR, whereas in general it refers to the choice between two processes. Similarly, the operator \cdot applied to two tests $\varphi, \psi \in B$ represents the logical AND, whereas in general it refers to the sequencing of processes. Furthermore, the constant 1 represents both TRUE and `skip`, whereas 0 represents both FALSE and `fail`. It can be easily seen from the axioms of Kleene algebras that this overloaded use of operators does not cause problems [452]. Intuitively, testing two conditions sequentially is equivalent to testing their conjunction, and a choice between two conditions reflects precisely the meaning of OR.

Example 4.3. Let us look again at the simple Example 3.14, where the WIS is used for ordering products. The corresponding story space is illustrated in Figure 4.1

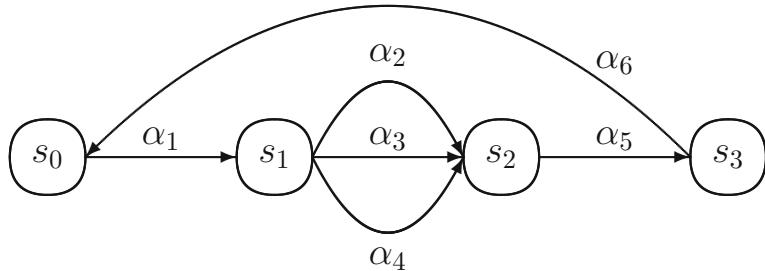


Fig. 4.1. Story space for Example 4.3 as a directed graph

Using the actions and conditions from Example 3.14 we can formalise the story space by the algebraic expression

$$(\alpha_1(\varphi_1\alpha_2\varphi_2 + \alpha_3\varphi_3 + \alpha_4\varphi_4)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^*,$$

which expresses exactly what we wanted to get. This expression contains twice optionality, which is expressed by the choice with `skip`. First, the action $\alpha_6 = \text{confirm_order}$ is optional, which just expresses that a user may decide to cancel an order. Then the whole iterated process is optional, which enables choosing `skip` for almost all iteration steps. This is equivalent to executing the sequence of ordering a product, selecting a payment method and confirming (or canceling) the order only finitely many times.

Now obviously the conditions appearing as guards and post-guards in a story algebra, form the set B of tests. Thus, a story algebra gives rise to a KAT.

Example 4.4. As a more sophisticated example take the loan application from Example 3.15. Then the algebraic expression

$$\begin{aligned} & \alpha_1((\varphi_0(\alpha_{21}\varphi_{13} + 1)\alpha_2 + \varphi_1\alpha_3(\alpha_5 + 1)\varphi_3 + \varphi_2\alpha_4(\alpha_6 + 1)\varphi_4)^*\varphi_5) \\ & (\alpha_7\varphi_6 + \alpha_{13}\varphi_7)(\varphi_6\alpha_8(\alpha_8 + 1)\alpha_9\alpha_{10}\alpha_{11}\alpha_{12}\varphi_8 \\ & + \varphi_7\alpha_8\alpha_8^*\alpha_{14}\alpha_{15}\alpha_{16}^*\alpha_{11}\alpha_{17}(\overline{\varphi_{12}}\alpha_{18}\alpha_{19})^*\varphi_{12}\alpha_{18}\varphi_9)\alpha_{20}(\varphi_{10} + \varphi_{11}) \end{aligned}$$

represents the plot using KAT operators.

Furthermore, we may define a *scene assignment* to a KAT by simply following the rules for the scene assignment in story algebras. That is, we obtain a partial mapping $\sigma : K \rightarrow \mathcal{S}$ with a set $\mathcal{S} = \{s_1, \dots, s_n\}$ of scenes as follows:

- For $p_1, p_2 \in K$ with $\sigma(p_1) = \sigma(p_2) = s$ or if one of the p_i is 1 or 0 or a test in B , then $\sigma(p_1 p_2) = s$.
- For $p_1, p_2 \in K$ with $\sigma(p_1) = \sigma(p_2) = s$ or if one of the p_i is 1 or 0 or a test in B , then $\sigma(p_1 + p_2) = s$.
- For $p \in K$ with $\sigma(p) = s$ we obtain $\sigma(p^*) = s$.

This leads to our definition of a many-sorted Kleene algebra with tests.

Definition 4.5. A *many-sorted Kleene algebra with tests* (MKAT) is a KAT $\mathcal{K} = (K, B, +, \cdot, ^*, \bar{\cdot}, 0, 1)$ together with a set $\mathcal{S} = \{s_1, \dots, s_n\}$ of scenes and a scene assignment $\sigma : K \rightarrow \mathcal{S}$ such that $p_1 p_2 = p_2 p_1$ holds for all $p_1, p_2 \in K$ with $\sigma(p_1) \neq \sigma(p_2)$.

From our discussion above it is clear that we can represent a story space by an element of the MKAT that is defined by the atomic actions, the tests and the scenes.

4.1.2 Customisation with Respect to Preferences and Goals

At the level of plots we can only address the customisation of functionality, i.e., we may restrict the plot according to preferences of users of a certain type and their goals. So the first step is to formalise the preferences associated with a user type. For this we may exploit that plots are represented by KAT expressions, so we may define such preferences by equations. In particular, we obtain the following types of equations.

Definition 4.6. The following are *preference conditions* associated with a user type or general conditions associated with a plot.

Preferences. An equation of the form $\varphi(p_1 + p_2) = \varphi p_1$ expresses that a user conditionally prefers process p_1 over process p_2 , where the condition is expressed by φ . The special case $\varphi = 1$ expresses an unconditional preference.

Precondition. An equation of the form $\bar{\varphi}\alpha = 0$ (or equivalently $\alpha = \varphi\alpha$) expresses that the condition φ is a precondition for the action α .

Postcondition. An equation of the form $\alpha\bar{\varphi} = 0$ (or equivalently $\alpha = \alpha\varphi$) expresses that the condition φ is a postcondition for the action α .

Invariance. An equation of the form $\alpha\varphi = \varphi\alpha$, which is equivalent to $\bar{\varphi}\alpha = \alpha\bar{\varphi}$ and to $\varphi\alpha\bar{\varphi} + \bar{\varphi}\alpha\varphi = 0$, expresses that the condition φ (and so its negation $\bar{\varphi}$) is invariant under the action α .

Exclusion. An equation of the form $\varphi\psi = 0$ expresses that the conditions φ and ψ exclude each other.

A *goal* of a user can be simply expressed by a postcondition ψ . Then customisation with respect to preferences and goals can be formalised by the following optimisation task.

Given a process $p \in K$ that represents a plot, and a set Σ of equations on K that represents (among other constraints) user preferences and a postcondition $\psi \in B$, we look for a minimal process $p' \in K$ with respect to the order on K such that $\Sigma \models p\psi = p'\psi$ holds.

That is, the resulting process p' is a *customisation* of p according to the *preferences* Σ and the *goal* formalised by ψ . We illustrate this kind of propositional reasoning with KATs by the following example.

Example 4.7. Let us continue Example 4.3 and localise the plot, i.e., customise with respect to the location of its users. Say, for an American user the bank transfer option should be invalid, which gives the equation $\alpha_3 = 0$. Note that this is a postcondition equation with the postcondition 0. Apart from this assume there is no particular goal given, i.e., the corresponding postcondition ψ is 1. If p denotes the KAT expression from Example 4.3 we can now compute

$$\begin{aligned} p\psi &= (\alpha_1(\varphi_1\alpha_2\varphi_2 + \alpha_3\varphi_3 + \alpha_4\varphi_4)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^* \\ &= (\alpha_1(\varphi_1\alpha_2\varphi_2 + 0 + \alpha_4\varphi_4)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^* \\ &= (\alpha_1(\varphi_1\alpha_2\varphi_2 + \alpha_4\varphi_4)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^* \\ &= p'\psi, \end{aligned}$$

i.e., the option to pay by bank transfer has been removed from the resulting plot p' .

As a second example consider a European user, for which the option to pay by cheque should be invalidated. So we obtain the equation $\alpha_4 = 0$. Furthermore, assume that this user prefers to pay by credit card, if this turns out to be a valid option. For this we use the equation $\varphi_1(\alpha_3 + \alpha_4) = 0$, which also implies $\varphi_1\alpha_3 = 0$. Let again $\psi = 1$. Now we compute

$$\begin{aligned} p\psi &= (\alpha_1(\varphi_1\alpha_2\varphi_2 + \alpha_3\varphi_3 + \alpha_4\varphi_4)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^* \\ &= (\alpha_1(\varphi_1\alpha_2\varphi_2 + \alpha_3\varphi_3)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^* \\ &= (\alpha_1(\varphi_1\alpha_2\varphi_2 + (\varphi_1 + \bar{\varphi}_1)\alpha_3\varphi_3)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^* \\ &= (\alpha_1(\varphi_1\alpha_2\varphi_2 + \varphi_1\alpha_3\varphi_3 + \bar{\varphi}_1\alpha_3\varphi_3)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^* \\ &= (\alpha_1(\varphi_1\alpha_2\varphi_2 + 0 + \bar{\varphi}_1\alpha_3\varphi_3)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^* \\ &= (\alpha_1(\varphi_1\alpha_2\varphi_2 + \bar{\varphi}_1\alpha_3\varphi_3)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^* \\ &= p'\psi, \end{aligned}$$

i.e., the option to pay by cheque has been removed from the resulting plot p' , and the option to pay by bank transfer has become subject to the condition $\bar{\varphi}_1$, i.e., the price must be out of range for credit card payment.

The approach taken in this Example 4.7 is in fact to start with the term $p\psi$ and to apply conditional rewrite rules to it. The rules arise from the preference conditions in Definition 4.6. Before formalising this term rewriting approach, let us first take a look at completeness and complexity issues.

Our formalisation above of the personalisation problem has led to an optimisation problem. However, in the theory of KATs so far only completeness, decidability and complexity of decision problems have been investigated. In other words, we only know something about problems of the form $\Sigma \models p = q$. Therefore, look at the following related decision problem:

Given processes $p, p' \in K$ such that p represents the plot of a story space and $p' \leq p$ holds, and a set Σ of equations on K that represents (among other constraints) user preferences and a postcondition $\psi \in B$, then decide, whether $\Sigma \models p\psi = p'\psi$ holds.

If the expression p that describes the story space is $*$ -free, there will be only finitely many expressions p' with $p' \leq p$. In this case, it is sufficient to investigate the related decision problem. If p involves the $*$ -operator, say we have a sub-expression q^* , it may be the case that replacing q^* by $(1 + q + q^2 + \dots + q^n)$ leads to some p' , for which $\Sigma \models p\psi = p'\psi$ holds. In this case, we may adopt the following pragmatic strategy:

1. Check whether for some n replacing q^* by $(1 + q + q^2 + \dots + q^n)$ leads to some p' with $\Sigma \models p\psi = p'\psi$.
2. If such an n exists, replace p by p' , thus remove the occurrence of the $*$ -operator.
3. If such an n does not exist, leave the $*$ -operator in p , and consider only those expressions $p' \leq p$, for which q^* is replaced by $(q')^*$ with $q' \leq q$.

While this strategy may miss out to find the optimal solution, it guarantees that we are able to reduce customisation to a decision problem.

For the equational theory of KATs we know from [456] that it is decidable, but PSPACE-complete. That is, we can decide in polynomial space, whether $p = q$ can be derived from the axioms of KATs. However, the decision problem we are dealing with is of the form $\Sigma \models p = q$ with a set of equations Σ , i.e., in the Horn theory of KATs. From [454] we already know that such problems are undecidable in general. Even if we reduce ourselves to Kleene algebras instead of KATs, we already get Σ_1^0 -completeness, i.e., the problem is in general recursively enumerable hard [451].

However, the problem we deal with is not the general decision problem for the Horn theory, as in our case Σ only contains equations that have a particular form. From [456] we know that we can reduce equations of the form $r = 0$ – these include equations obtained from pre- and postconditions, exclusion conditions, and invariance equations – to the equational theory. Instead of showing $r = 0 \rightarrow p\psi = p'\psi$ we can equivalently show

$$p\psi + uru = p'\psi + uru,$$

where u is a “universal” process. That is $u = (\alpha_1 + \dots + \alpha_m)^*$, where the α_i are all the atomic actions that are not tests – it is easy to see (and crucial for the proof) that $p \leq u$ holds for all processes p . So we can remove all equations of the form $r = 0$ from Σ and enrich the equation that is to be derived instead.

Example 4.8. Let us look again at the European user in Example 4.7. In this case we had two equations $\alpha_4 = 0$ and $\varphi_1(\alpha_3 + \alpha_4) = 0$. We have to show that we can derive from this $p = p'$ with

$$p = (\alpha_1(\varphi_1\alpha_2\varphi_2 + \alpha_3\varphi_3 + \alpha_4\varphi_4)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^*$$

and

$$p' = (\alpha_1(\varphi_1\alpha_2\varphi_2 + \bar{\varphi}_1\alpha_3\varphi_3)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^*$$

According to the result above it is equivalent to derive $p + u\alpha_4u + u\varphi_1\alpha_3u = p' + u\alpha_4u + u\varphi_1\alpha_3u$ from the axioms of KATs. We will briefly show how this can be proven in two steps using also

$$p_1 = (\alpha_1(\varphi_1\alpha_2\varphi_2 + \alpha_3\varphi_3)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^*.$$

We first show $p + u\alpha_4u = p_1 + u\alpha_4u$, then $p_1 + u\varphi_1\alpha_3u = p' + u\varphi_1\alpha_3u$. For the first step we observe that we can write $p_1 = \beta_2^*$, while we have $p = (\alpha_1\alpha_4\beta_1 + \beta_2)^*$. With this we can derive

$$\begin{aligned} \beta_2^* + u\alpha_4u &= p_1 + u\alpha_4u \leq p + u\alpha_4u = (\alpha_1\alpha_4\beta_1 + \beta_2)^* + u\alpha_4u \leq \\ (u\alpha_4u + \beta_2)^* + u\alpha_4u &= \beta_2^* + \beta_2^*u\alpha_4u((u\alpha_4u)^*\beta_2^*)^* + u\alpha_4u \leq \beta_2^* + u\alpha_4u. \end{aligned}$$

As the first and the last process in this inequality chain are the same, we must have equality everywhere. For the second step we proceed analogously, now using $p_1 = (\alpha_1\varphi_1\alpha_3\beta_3 + \beta_4)^*$ and $p' = \beta_4^*$.

4.1.3 Conditional Term Rewriting on KATs

Let us return to our original minimisation problem. As indicated in Example 4.7 it seems to be a promising idea to use rewrite rules $\lambda \rightarrow \rho$, until we obtain an irreducible term of the form $p'\psi$, i.e., none of the rewrite rules can be applied anymore. If we guarantee that each application of a rewrite rule leads to a smaller term with respect to the order \leq , this irreducible term must be minimal and p' must be the solution to the personalisation problem.

The Knuth-Bendix-algorithm from [431] provides the necessary technique and theoretical justification for this approach. Basically, it is a term-rewriting approach that is coupled with so-called “critical-pair completion” (CPC), a method that permits the generation of new rewriting rules during rewriting. This generation of new rules will be necessary, as we cannot guarantee completeness in general.

In order to exploit this approach for our purposes here, i.e., rewriting in KATs with personalisation equations, we need a few extensions:

- For KATs we need an order-sorted approach, as we have a sort B of Booleans, a sort K for the Kleene algebra terms, and $B \leq K$. This is simply added to the signature.

- We have to deal with conditional rewrite rules in order to capture the axioms of the star-operator. For this we use co-routines, which validate the preconditions of a rule in a separate rewriting process.
- We have to deal with commutativity, for which we take the pragmatic approach to use simultaneous rewriting considering all alternatives at the same time.

However, by exploiting this Knuth-Bendix-type approach there is no need to eliminate equations of the form $r = 0$, as these can just be added to the set of available rewrite rules. Similarly, the suggested trick for dealing with conditions preferences is also not needed. In principle it is even possible to exploit further types of personalisation equations extending those from Definition 4.6. However, we cannot guarantee that the CPC procedure will terminate.

As standard in order-sorted algebraic specifications we take two *sorts* B and K ordered by $B \leq K$, and the following (nullary, unary, binary) *operators*:

$$0, 1 : \rightarrow B \quad +, \cdot : K \ K \rightarrow K \quad * : K \rightarrow K \quad \bar{} : B \rightarrow B$$

Using these sorts and operators we can define *terms* in the usual way:

- Each variable of sort B or K is also a term of sort B or K , respectively.
- 0 and 1 are terms of sort B .
- If t_1 , t_2 and t_3 are terms of sort K , then also $t_1 + t_2$, $t_1 \cdot t_2$, and t_3^* .
- If t is a term of sort B , then also \bar{t} .
- Each term of sort B is also a term of sort K .

Definition 4.9. A *rewrite rule* is an expression of the form $\lambda \rightsquigarrow \varrho$ with terms λ and ϱ of the same sort, such that the variables on the right hand side ϱ are a subset of the variables on the left hand side λ . A *conditional rewrite rule* is an expression of the form $t_1 = t_2 \rightarrow \lambda \rightsquigarrow \varrho$, in which in addition the terms t_1 and t_2 contain the same variables and these form a superset of the set of variables in the left hand side term λ .

The application of a rewrite rule $\lambda \rightsquigarrow \varrho$ to a term t is standard: If t contains a subterm t' that can be matched with λ , i.e., there is a substitution θ such that the application of θ to λ results in t' (denoted $\theta.\lambda = t'$), then replace t' in t by $\theta.\varrho$.

The application of a conditional rewrite rule $t_1 = t_2 \rightarrow \lambda \rightsquigarrow \varrho$ to a term t is defined analogously. Precisely, if t contains a subterm t' that can be matched with λ , i.e., there is a substitution θ such that the application of θ to λ results in t' (denoted $\theta.\lambda = t'$), then replace t' in t by $\theta.\varrho$. However, in this case we have to show that $\theta.t_1 = \theta.t_2$ holds for the substitution θ . For this we start a separate term-rewriting process that aims at showing $\theta.t_1 \rightsquigarrow \dots \rightsquigarrow \theta.t_2$. We call this separate rewriting process a *co-routine*, because we can run it in parallel to the main rewriting process. The risk is of course that if we fail

to verify $\theta.t_1 = \theta.t_2$, then we have to backtrack to t for the main rewriting process.

In order to exploit term-rewriting for the personalisation problem we formulate the axioms of KATs and the personalisation equations as (conditional) rewrite rules, then start with $p\psi$ and apply the rules until we finally obtain a term of the form $p'\psi$ to which no more rule can be applied. Note that $p\psi$ is closed, i.e., it does not contain variables, so during the whole rewriting process we will only have to deal with closed terms.

We use p, q, r, \dots (if needed with additional indices) as *variables* of sort K , and a, b, c, \dots (also with indices) as *variables* of sort B . Then we use the following general (conditional) rewrite rules:

$$\begin{array}{ll}
 p + (q + r) \rightsquigarrow (p + q) + r & p(qr) \rightsquigarrow (pq)r \\
 p + p \rightsquigarrow p & p + 0 \rightsquigarrow p \\
 p1 \rightsquigarrow p & 1p \rightsquigarrow p \\
 p0 \rightsquigarrow 0 & 0p \rightsquigarrow 0 \\
 p(q + r) \rightsquigarrow pq + pr & (p + q)r \rightsquigarrow pr + qr \\
 1 + pp^* \rightsquigarrow p^* & 1 + p^*p \rightsquigarrow p^* \\
 pq + q = q \rightarrow p^*q + q \rightsquigarrow q & qp + q = q \rightarrow qp^* + q \rightsquigarrow q \\
 p + q \rightsquigarrow q + p & ab \rightsquigarrow ba \\
 a\bar{a} \rightsquigarrow 0 & \bar{a}a \rightsquigarrow 0 \\
 a + \bar{a} \rightsquigarrow 1 & \bar{a} + a \rightsquigarrow 1
 \end{array}$$

As an extension we apply these rewriting rules $\lambda \rightsquigarrow \varrho$ not only in one direction, but we also use the reverse rules $\varrho \rightsquigarrow \lambda$ (similar for conditional rewrite rules). In addition, the customisation equations from Definition 4.6 give rise to further rewrite rules:

- A conditional preference equation gives rise to a rule of the form $a(p+q) \rightsquigarrow ap$.
- A precondition gives rise to a rule of the form $\bar{a}p \rightsquigarrow 0$.
- A postcondition gives rise to a rule of the form $p\bar{a} \rightsquigarrow 0$.
- An invariance condition gives rise to a rule of the form $ap\bar{a} + \bar{a}pa \rightsquigarrow 0$.
- An exclusion condition gives rise to a rule of the form $ab \rightsquigarrow 0$.

Thus, let Σ denote a set of equations on a KAT $\mathcal{K} = (K, B, +, \cdot, *, \bar{}, 0, 1)$. Then Σ defines an equivalence relation \sim on K in the usual way, i.e., $p \sim q$ holds iff $p = q$ is implied by Σ and the axioms for KATs in Definitions 4.1 and 4.2.

Let K/Σ denote the set of equivalence classes $[p]$ under \sim with $p \in K$, and B/Σ the subset $\{[b] \mid b \in B\}$. Define the KAT operations on K/Σ in the natural way, i.e., $[p] + [q] = [p + q]$, $[p][q] = [pq]$, $[p]^* = [p^*]$, $0 = [0]$, and $1 = [1]$, and the complementation on B/Σ by $\bar{[b]} = [\bar{b}]$.

Proposition 4.10. *If $\mathcal{K} = (K, B, +, \cdot, *, \bar{\cdot}, 0, 1)$ is a KAT and Σ a set of equations on K , then the quotient $\mathcal{K}/\Sigma = (K/\Sigma, B/\Sigma, +, \cdot, *, \bar{\cdot}, 0, 1)$ is a KAT.*

The basic rewriting procedure from [431] uses only unconditional rewrite rules, for which a Church-Rosser property can be shown. This result permits a rather “careless” application of rules. If more than one rule can be applied, it does not matter which is selected first. This does not apply in our case, as the rules arising from commutativity and parallelism may lead to loops, thus have to be handled with care. Furthermore, the presence of the $*$ -operator resulted in conditional rewrite rules, for which we need an extension.

For a conditional rewrite rule $t_1 = t_2 \rightarrow \lambda \rightsquigarrow \varrho$ whenever λ matches a subterm t' using the substitution θ , the equation $\theta.t_1 = \theta.t_2$ has to be verified. That is, when we start a *co-routine*, i.e., a separate rewriting process for $\theta.t_1$ and $\theta.t_2$ to verify this equation. While this co-routine is processed, the rewriting of the original term can be continued until the condition is verified or disproved. In the latter case backtracking has to be applied. Example 4.12 shows the application of co-routines.

To avoid loops, when we are given a term t , we apply all these commutative rules simultaneously and obtain a list of terms, say t_1, \dots, t_n , to which the rewriting process is applied. Reversing a commutativity rule would only produce terms that are already in the list. These terms can be discarded.

Example 4.11. Applying this procedure, the equations in Example 4.7 represent indeed a rewriting chain, i.e., we can replace all equality symbols by \rightsquigarrow . In detail, we start with

$$p\psi = (\alpha_1(\varphi_1\alpha_2\varphi_2 + \alpha_3\varphi_3 + \alpha_4\varphi_4)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^*$$

Here we can consider the subterm α_3 , which matches with the left hand side of the precondition rule $\alpha_3 \rightsquigarrow 0$ with an empty substitution, i.e., we obtain

$$p\psi \rightsquigarrow (\alpha_1(\varphi_1\alpha_2\varphi_2 + 0 \cdot \varphi_3 + \alpha_4\varphi_4)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^*$$

In this term the subterm $0 \cdot \varphi_3$ matches the left hand side of the rule $0p \rightsquigarrow 0$ with the substitution $\theta = \{p/\varphi_3\}$, so we can further rewrite the term to

$$(\alpha_1(\varphi_1\alpha_2\varphi_2 + 0 + \alpha_4\varphi_4)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^*$$

Here, the subterm $0 + \alpha_4\varphi_4$ matches the left hand side of rule $0 + p \rightsquigarrow p$ with the substitution $\theta = \{p/\alpha_4\varphi_4\}$, so we further rewrite the term to

$$(\alpha_1(\varphi_1\alpha_2\varphi_2 + \alpha_4\varphi_4)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^*$$

which was our final result $p'\psi$.

Note, however, that our algorithm would compute much more branches simultaneously, none of which will lead to a different result.

For the second term in Example 4.7 we had

$$p\psi = (\alpha_1(\varphi_1\alpha_2\varphi_2 + \alpha_3\varphi_3 + \alpha_4\varphi_4)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^*,$$

for which we choose the subterm α_4 . So we can apply $\alpha_4 \rightsquigarrow 0$ to obtain the term

$$(\alpha_1(\varphi_1\alpha_2\varphi_2 + \alpha_3\varphi_3 + 0\varphi_4)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^*.$$

Next we apply the rule $0p \rightsquigarrow 0$ with the substitution $\theta = \{p/\varphi_4\}$, which results in the term

$$(\alpha_1(\varphi_1\alpha_2\varphi_2 + \alpha_3\varphi_3 + 0)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^*.$$

The subterm $\alpha_3\varphi_3 + 0$ matches the left hand side of the rule $p + 0 \rightsquigarrow p$ with the substitution $\theta = \{p/\alpha_3\varphi_3\}$, so we obtain now

$$(\alpha_1(\varphi_1\alpha_2\varphi_2 + \alpha_3\varphi_3)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^*.$$

So far, we applied the same rules as in the first example. Now the subterm α_3 matches the left hand side of rule $p \rightsquigarrow 1 \cdot p$ with the substitution $\theta = \{p/\alpha_3\}$, so the corresponding rewrite step leads to the term

$$(\alpha_1(\varphi_1\alpha_2\varphi_2 + 1\alpha_3\varphi_3)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^*.$$

The subterm 1 matches the left hand side of the rule $1 \rightsquigarrow a + \bar{a}$ with the substitution $\theta = \{a/\varphi_1\}$, which gives us the term

$$(\alpha_1(\varphi_1\alpha_2\varphi_2 + (\varphi_1 + \bar{\varphi}_1)\alpha_3\varphi_3)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^*.$$

Using the substitution $\theta = \{p/\varphi_1, q/\bar{\varphi}_1, r/\alpha_3\varphi_3\}$ the subterm $(\varphi_1 + \bar{\varphi}_1)\alpha_3\varphi_3$ matches the left hand side of the rule $(p + q)r \rightsquigarrow pr + qr$, so by applying this rule to this subterm we obtain the term

$$(\alpha_1(\varphi_1\alpha_2\varphi_2 + \varphi_1\alpha_3\varphi_3 + \bar{\varphi}_1\alpha_3\varphi_3)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^*.$$

Next we can apply the precondition rule $\varphi_1\alpha_3 \rightsquigarrow 0$ using the subterm $\varphi_1\alpha_3$ and an empty substitution. This gives us

$$(\alpha_1(\varphi_1\alpha_2\varphi_2 + 0\varphi_3 + \bar{\varphi}_1\alpha_3\varphi_3)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^*.$$

Finally, we first apply the rule $0p \rightsquigarrow 0$ using the subterm $0\varphi_3$ and the substitution $\theta = \{p/\varphi_3\}$, then the rule $p + 0 \rightsquigarrow p$ for the subterm $\varphi_1\alpha_2\varphi_2$ and the substitution $\theta = \{p/\varphi_1\alpha_2\varphi_2\}$ to obtain the final result

$$p'\psi = (\alpha_1(\varphi_1\alpha_2\varphi_2 + \bar{\varphi}_1\alpha_3\varphi_3)\alpha_5(\alpha_6\varphi_5 + 1) + 1)^*.$$

Example 4.12. The previous example is straightforward, as it does not require conditional term rewriting. So let us now show how co-routines could come into the picture. Assume that a subterm of a given KAT expression has the form $(\bar{\varphi}p)^*\varphi$, representing a while-loop: as long as condition φ is not satisfied we execute the process p . Now assume that p^2 has the postcondition φ , i.e.,

$p^2\bar{\varphi} = 0$ holds. So we know that p has to be executed at most twice, i.e., the loop expression should be rewritten to $(1 + \bar{\varphi}p + (\bar{\varphi}p)^2)\varphi$.

Starting with $(\bar{\varphi}p)^*\varphi$ we can apply the rule $p^* \rightsquigarrow 1 + pp^*$ several times to obtain the term

$$(1 + \bar{\varphi}p + \bar{\varphi}p\bar{\varphi}p + \bar{\varphi}p\bar{\varphi}p\bar{\varphi}p(\bar{\varphi}p)^*)\varphi.$$

With the shortcut $q = 1 + \bar{\varphi}p + \bar{\varphi}p\bar{\varphi}p$, we see that we can rewrite $q + \bar{\varphi}p\bar{\varphi}p\bar{\varphi}p(\bar{\varphi}p)^*)\varphi$ simultaneously to $q + \bar{\varphi}p\bar{\varphi}p(\bar{\varphi}p)^*$, $q + \bar{\varphi}p(\bar{\varphi}p)^*$, and $q + (\bar{\varphi}p)^*$, so we can rewrite

$$\begin{aligned} & (q + \bar{\varphi}p\bar{\varphi}p\bar{\varphi}p(\bar{\varphi}p)^*)\varphi \\ \rightsquigarrow & (q + (\bar{\varphi}p)^* + \bar{\varphi}p(\bar{\varphi}p)^* + \bar{\varphi}p\bar{\varphi}p(\bar{\varphi}p)^*)\varphi \\ \rightsquigarrow & (q(\bar{\varphi}p)^* + q)\varphi \end{aligned}$$

This can be further rewritten to $q\varphi$, i.e., to $(1 + \bar{\varphi}p + \bar{\varphi}p\bar{\varphi}p)\varphi$ as desired using the conditional rule $qp' + q = q \rightarrow q(p')^* + q \rightsquigarrow q$ and the substitution $\{p'/\bar{\varphi}p\}$, if we can verify

$$(1 + \bar{\varphi}p + \bar{\varphi}p\bar{\varphi}p)\bar{\varphi}p + 1 + \bar{\varphi}p + \bar{\varphi}p\bar{\varphi}p = 1 + \bar{\varphi}p + \bar{\varphi}p\bar{\varphi}p.$$

For this we have to start a co-routine, in which the postcondition rule $p^2\bar{\varphi} \rightsquigarrow 0$ can be applied to yield the desired result. In more detail, we first rewrite the left hand side of this equation to

$$1 + \bar{\varphi}p + \bar{\varphi}p\bar{\varphi}p + \bar{\varphi}p\bar{\varphi}p\bar{\varphi}p$$

using rules such as $(p+q)r \rightsquigarrow pr + qr$ and $p+p \rightsquigarrow p$. Then our postcondition rule implies $(\bar{\varphi}p)^2\bar{\varphi} \rightsquigarrow 0$, so we can further rewrite the term to $1 + \bar{\varphi}p + \bar{\varphi}p\bar{\varphi}p + 0$ and thus to $1 + \bar{\varphi}p + \bar{\varphi}p\bar{\varphi}p$ as desired.

It may happen that the rewriting process gets stuck in the sense that we may rewrite a term t to terms t_1 and t_2 , but neither can we rewrite t_1 to t_2 nor the other way round. In this case t_1 and t_2 represent a *critical pair*, which allows us to add a new rewrite rule $t_1 \rightsquigarrow t_2$ (and its reverse) and continue the rewriting process. This process is called critical-pair completion.

Critical-pair completion has been introduced by D. Knuth and P. Bendix in [431]. Their seminal work contains the strict mathematical foundation of this approach including the justification of its correctness. In a nutshell, this justification is quite simple, as all rewriting rules represent (directed) equations. Whatever is substituted for the variables in a rewrite rule, its left and right hand sides are equal terms. In our case, the rewrite rules correspond to the equations that define KATs, and those that arise from user preferences. So, if a term t can be rewritten to two other terms t_1 and t_2 , it will be equal to both of them, hence t_1 and t_2 are equal and adding the corresponding rewrite rules does not change the equational theory. However, as we cannot assume completeness, the additional rules may help to derive equations that cannot be derived from the initial set of rules. In other words, while there

may be no complete set of rules, critical-pair completion enlarges the set of derivable equations. This process is successfully applied in computer algebra, computational geometry and other areas of applied symbolic computation.

In principle, critical-pair completion can also be applied, when t_1 and t_2 arise from conditional rewriting rules, in which case we would obtain a new conditional rewriting rule. Pragmatically speaking it would be better to wait for the successful termination of the co-routine before adding such conditional rules.

The term rewriting approach depends on user types in the sense that the available rewrite rules are determined by a user type. However, we define a user type as a subset of $gr(\Delta)$, the set of user profiles. Each user profile is defined by values for the dimensions in Δ . As an alternative we can therefore make all rewrite rules dependent on values of dimensions, i.e., our rules – we may restrict ourselves to conditional preference equations – take the form

$$\delta_{i_1} = v_{i_1} \wedge \cdots \wedge \delta_{i_k} = v_{i_k} \rightarrow a(p + q) \rightsquigarrow ap,$$

with $\delta_{i_j} \in \Delta$ and $v_{i_j} \in dom(\delta_{i_j})$ for $j = 1, \dots, k$. In this way we can dispense with setting up user types, as we only have to check the values of the dimensions. If we know the values of the dimensions in advance, the actual rewrite procedure does not change very much, as a simple preprocessing of the rules to determine which of them may be applied will be sufficient.

However, the rewrite process changes, if the values of the dimensions only become available during system usage. In this case we may add rules to the set of available rules. As additional rules may further simplify the plot, the situation may arise that a user has already executed an action that would be discarded from his or her personal plot, if the user type had been fully known. The solution to this problem is backtracking, i.e., discarding the latest actions of a user and considering only a prefix of his or her actual story that is compatible with the personalised plot.

4.1.4 Church Rosser Property

The outlined term rewriting process is indeed highly non-deterministic, but we wish to obtain a unique final result. For this we first have to ensure that the process terminates. If this can be done, then the second problem is to ensure confluence, i.e., guaranteeing that the result is independent from the choice of rewrite rules and their order. We will investigate these two problems in this section.

In [722] the idea was launched to approach personalisation by term rewriting on KATs. Indeed, the plot of a storyboard can be easily written as a KAT expression, and preference rules give rise to equations as exemplified in the previous section. Then the problem is to find a “simplest” representative $p' \in K$ of an equivalence class $[p] \in K/\Sigma$.

Equations like the ones shown in the previous section show indeed a preferred direction. We would rather see the left hand side replaced by the right

hand side than the other way round. That is, we could use equations as rewrite rules $\ell \rightsquigarrow r$. This was exemplified in [722, 727] and further elaborated in [734]. It may still be necessary to use rules such as $1 \rightsquigarrow b + \bar{b}$ for some $b \in B$, but for the moment we are primarily interested in termination.

For this we take up the old idea from [190] to define a well-founded order on terms in such a way that the left hand side of the rewrite rules will always be larger than the right hand side. In particular, we will look at an order defined by formal power series. Thus, our first step will be to look for ways to associate a formal power series with the expressions in a KAT $\mathcal{K} = (K, B, +, \cdot, *, \bar{}, 0, 1)$.

In order to do so, we start with the assumption that we are given a set $A = \{p_1, \dots, p_k\}$ of atomic processes and a set $P = \{b_1, \dots, b_\ell\}$ of propositional atoms with $A \cap P = \emptyset$. Let B be generated out of $P \cup \{0, 1\}$ using conjunction \cdot , disjunction $+$, and negation $\bar{}$. Then let K be generated out of $A \cup B \cup \{0, 1\}$ using choice $+$, sequence \cdot , and iteration $*$. More precisely, we have to consider K/Σ with Σ being the KAT axioms from Definitions 4.1 and 4.2, but we ignore this little subtlety.

Note that we actually restrict our attention to finitely generated KATs, which will help us to circumvent the undecidability and complexity problems associated with KATs in general [456].

We even extend the Boolean algebra B by enlarging its base P . The idea is that for $p \in A$ and $b \in P \cup \{0\}$ we add the weakest preconditions $[p]b$ to P . The intended meaning is if p starts in a state satisfying $[p]b$, for which it is defined and terminating, then the resulting state will satisfy b . Formally, $[p]b$ is the largest solution x of $xpb = 0$. For simplicity let us stay with the notation P , B and K .

Definition 4.13. A *monom* over A of degree m is an expression of the form $q_1 \dots q_m$ with all $q_i \in A$. A *formal power series* over A and P is an expression of the form

$$\sum_{i=0}^{\infty} \sum_{\sigma: \{1, \dots, i\} \rightarrow \{1, \dots, k\}} b_{\sigma(1) \dots \sigma(i)} p_{\sigma(1)} \dots p_{\sigma(i)}$$

with coefficients $b_{\sigma(1) \dots \sigma(i)} \in B$. Let $\mathcal{FPS}(K)$ denote the set of formal power series over K .

Note that the Boolean algebra B in this definition contains all the preconditions, and the $p_{\sigma(1)} \dots p_{\sigma(i)}$ are indeed monoms of degree i . Furthermore, assuming an order \leq on A , this can be canonically extended to a total order on monoms, in which monoms of lower degree precede all those of higher degree.

We have to ensure that these basic preconditions are sufficient. For this we shift Boolean conditions in front of processes leading to the following proposition.

Proposition 4.14. *With the definitions above each KAT expression $p \in K$ can be represented by a formal power series $\llbracket p \rrbracket \in \mathcal{FPS}(K)$.*

Proof. Looking at composed propositions we have

$$[p](b_1 b_2) = ([p]b_1)([p]b_2) \quad \text{and} \quad [p]\bar{b} = \overline{[p]0[p]\bar{b}},$$

which together with $b_1 + b_2 = \overline{\bar{b}_1 \bar{b}_2}$ allows us to replace composed propositions by simply ones in P .

For complex processes in K we obtain $[1]b = b$, $[0]b = 0$, $[pq]b = [p][q]b$, and $[p+q]b = ([p]b)([q]b)$. In addition, $[p^*]b$ is the smallest solution x of $x = b[p]x$, but this may not be needed, as we can use the formal power series $p^* = 1 + p + p^2 + \dots = \sum_{i=0}^{\infty} p^i$ for our purposes, which will result in

$$[p^*]b = \sum_{i=0}^{\infty} [p^i]b = \sum_{i=0}^{\infty} \underbrace{[p] \dots [p]}_{i \text{ times}} b .$$

According to the definition of weakest precondition we have $[p]bp = pb$, so we obtain that all propositional conditions can be shifted to the front. \square

The second step towards our goal of terminating term rewriting consists of defining a well-founded partial order on K such that for rewrite rules $\ell \rightsquigarrow r$ we are interested in we obtain $\ell \geq r$. This suffices to guarantee termination [190]. We will concentrate on fps orders.

Definition 4.15. An *fps order* \leq on K is defined by a valuation $\nu : B \rightarrow \mathbb{R}$. For KAT expressions $p_1, p_2 \in K$ with

$$\llbracket p_j \rrbracket = \sum_{i=0}^{\infty} \sum_{\sigma : \{1, \dots, i\} \rightarrow \{1, \dots, k\}} b_{\sigma(1) \dots \sigma(i)}^{(j)} p_{\sigma(1)} \dots p_{\sigma(i)}$$

we define $p_1 \leq p_2$ iff $p_1 = p_2$ or there exists some sequence α such that $\nu(b_\alpha^{(1)}) < \nu(b_\alpha^{(2)})$ and $\nu(b_\beta^{(1)}) = \nu(b_\beta^{(2)})$ hold for all $\beta < \alpha$.

For instance, we could write $b \in B$ as a disjunction of minterms over P , and use this to define $\nu(b)$ using the equations

$$\nu(b) = \frac{1}{2} = \nu(\bar{b}) \text{ for } b \in P, \quad \nu(b_1 b_2) = \nu(b_1) \cdot \nu(b_2), \quad \nu(b_1 + b_2) = \nu(b_1) + \nu(b_2)$$

The fps order defined by this valuation guarantees that all rewrite rules obtained from the preference equations above will have the desired property. More generally, we can now characterise some classes of rewrite rules that are of particular interest by the property that we obtain a descending sequence of KAT expressions with respect to some fps order.

Definition 4.16. A set \mathcal{R} of rewrite rules $\ell \rightsquigarrow r$ derived from a set Σ of equations $\ell = r$ on K is *safe* iff there exists an fps order \leq on K such that $r \leq \ell$ holds for all rewrite rules in $\ell \rightsquigarrow r \in \mathcal{R}$, and for all rewrite rules in $\ell \rightsquigarrow r \in \mathcal{R}$ the right hand side r is a polynomial.

Note that we do not require $r < \ell$, because we can only have finitely many equivalent terms. The following proposition states that indeed a safe set of rewrite rules will guarantee termination.

Proposition 4.17. *If \mathcal{R} is a safe set of rewrite rules on K , then term rewriting with \mathcal{R} will always terminate.*

Proof. As we have $r \leq \ell$ for all rewrite rules $\ell \rightsquigarrow r$ term rewriting produces a decreasing sequence of expressions p_0, p_1, p_2, \dots . There cannot be infinitely many p_i with the same valuation, so the rewriting terminates with an expression of minimal valuation. \square

Note that our definition of fps-order does not require the full expansion of the power series representation, which would be infeasible. Furthermore, the termination result assumes that we apply rewrite rules to finite expressions, not to the expanded power series. The restriction that right hand sides of rewrite rules must be polynomials could thus be relaxed by the requirement that rules not satisfying this property can only be applied finitely many times.

Besides termination we are interested in a unique result of the rewriting process, for which we have to investigate confluence. Formally, (conditional) term rewriting defines a binary relation \rightsquigarrow on terms. Let \rightsquigarrow^* denote its reflexive, transitive closure.

Definition 4.18. A (conditional) term rewriting system is *confluent* iff for all terms t, t_1, t_2 whenever $t \rightsquigarrow^* t_1$ and $t \rightsquigarrow^* t_2$ hold, there exists a term t' with $t_1 \rightsquigarrow^* t'$ and $t_2 \rightsquigarrow^* t'$.

A (conditional) term rewriting system is *locally confluent* iff for all terms t, t_1, t_2 whenever $t \rightsquigarrow t_1$ and $t \rightsquigarrow t_2$ hold, there exists a term t' with $t_1 \rightsquigarrow^* t'$ and $t_2 \rightsquigarrow^* t'$.

A terminating and confluent (conditional) term rewriting system satisfies the *Church-Rosser property*, i.e., for each term t there exists a unique term t^* with $t \rightsquigarrow^* t^*$ such that no rewrite rule is applicable to t^* . The term t^* is usually called the *normal form* of t . In order context of WIS customisation, in which t and t^* would represent plots, we call t^* the *customisation* of t with respect to the preference rules used for term rewriting.

For terminating term rewriting systems the notions of confluence and local confluence coincide. Therefore, assuming that we use an fps-order we may restrict our attention to local confluence. Nevertheless, it is easy to see that the application of the rewrite rules as defined in the previous section will not be locally confluent. In order to circumvent this problem we suggest adopting critical pair completion from the approach of Knuth and Bendix [431].

Definition 4.19. In case $t \rightsquigarrow^* t_1$ and $t \rightsquigarrow^* t_2$ hold and neither t_1 nor t_2 permit a further application of rewrite rules, the pair (t_1, t_2) is called *critical*.

The idea of critical pair completion is to add a new rewrite rule $t_1 \rightsquigarrow t_2$ (and its reverse) and continue the rewriting process. In the presence of an fps-order, however, we would check first if $t_1 \leq t_2$ holds or $t_2 \leq t_1$. In the former case we would add $t_2 \rightsquigarrow t_1$, in the latter one $t_1 \rightsquigarrow t_2$ to the set of rewrite rules. Obviously, in this way we enforce confluence and thus the Church-Rosser property.

Proposition 4.20. *If a safe set \mathcal{R} of rewrite rules on K is combined with critical pair completion, term rewriting with \mathcal{R} will satisfy the Church-Rosser property.*

Proof. Due to Proposition 4.17 term rewriting with \mathcal{R} will terminate. If there is more than one possible way to terminate, we will have a critical pair and thus be able to continue the rewriting process with an extended set of rules. Due to the well-foundedness of the fps-order the process cannot continue forever, as we either terminate with 0 or with a unique minimal expression. \square

Example 4.21. In Example 4.7 we actually applied the rewrite step $\alpha_3 \rightsquigarrow (\varphi_1 + \bar{\varphi}_1)\alpha_3$. This can be rewritten to $\varphi_1\alpha_3 + \bar{\varphi}_1\alpha_3$ and further to $\bar{\varphi}_1\alpha_3$ using the preference rule $\varphi_1\alpha_3 = 0$. On the other hand we could undo the introduction of the disjunction and rewrite the term to α_3 . Critical pair completion would thus add the rewrite rule $\alpha_3 \rightsquigarrow \varphi_1\alpha_3$ and thus enforce the Church-Rosser property.

4.2 Compatibility of Preference Rules with Deontic Constraints

We explained that the concept of actor in storyboarding does not only refer to user profiles and associated preferences but also to user roles and associated obligations and rights. For these we can use a propositional deontic logic [727]. Now the question arises how to coordinate the deontic constraints of a role with preferences of a user in that role (according to some profile).

Let us first introduce the logical language \mathcal{L} for obligations and rights.

Definition 4.22. The set of deontic formulae is the smallest set \mathcal{L} with the following properties:

- All propositional atoms are also atoms of \mathcal{L} .
- If α is an action on scene s and r is a role associated with s , then $\mathbf{O} do(r, \alpha)$ is an atom of \mathcal{L} .
- If α is an action on scene s and r is a role associated with s , then $\mathbf{P} do(r, \alpha)$ is an atom of \mathcal{L} .
- If α is an action on scene s and r is a role associated with s , then $\mathbf{F} do(r, \alpha)$ is an atom of \mathcal{L} .
- If α is an action on scene s and r is a role associated with s , then $do(r, \alpha)$ is an atom of \mathcal{L} .

- For $\varphi, \psi \in \mathcal{L}$ we also have $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \Rightarrow \psi$ and $\varphi \Leftrightarrow \psi$ are also formulae in \mathcal{L} .

The informal meaning of the operators \mathbf{O} , \mathbf{P} and \mathbf{F} is standard: $\mathbf{O} \text{ } do(r, \alpha)$ means that an actor with role r is obliged to perform action α , $\mathbf{P} \text{ } do(r, \alpha)$ means that an actor with role r is permitted to perform action α , and $\mathbf{F} \text{ } do(r, \alpha)$ means that an actor with role r is forbidden to perform action α . The meaning of the atom $do(r, \alpha)$ is that an actor in role r will actually perform action α .

Example 4.23. Recall from our simple Example 4.3 that α_2 is the payment_by_card action, and α_4 is the payment_by_cheque action. The condition $\varphi_1 = \text{price_in_range}$ expresses that the price of the selected product lies within a range permitted for credit card payment. Then the deontic formula

$$\neg\varphi_1 \Rightarrow \mathbf{F} \text{ } do(\text{new_customer}, \alpha_4)$$

expresses that if the price of the selected product is outside the specified range for credit card payment, a new customer may also not pay by cheque. The deontic constraint

$$\mathbf{P} \text{ } do(\text{trusted_customer}, \alpha_2) \wedge \mathbf{P} \text{ } do(\text{trusted_customer}, \alpha_4)$$

expresses that a trusted customer is allowed to unconditionally pay by credit card or cheque.

Let us briefly discuss the semantics of our logic, for which we use status sets.

Definition 4.24. A *status set* of \mathcal{L} is a set \mathcal{S} of atoms satisfying $\mathbf{P} \text{ } do(r, \alpha) \in \mathcal{S}$ iff $\mathbf{F} \text{ } do(r, \alpha) \notin \mathcal{S}$.

If Σ is a set of formulae in \mathcal{L} , then a status set \mathcal{S} is *feasible* iff it satisfies Σ in the usual propositional sense, i.e., \mathcal{S} determines the truth values of the atoms, and the usual interpretation for negation, conjunction and disjunction applies. \mathcal{S} is *closed* iff we have $\mathbf{O} \text{ } do(r, \alpha) \in \mathcal{S} \Rightarrow do(r, \alpha) \in \mathcal{S}$ and $do(r, \alpha) \in \mathcal{S} \Rightarrow \mathbf{P} \text{ } do(r, \alpha) \in \mathcal{S}$.

Obviously, we can use the rules defining closed status sets to build the closure of any status set.

Definition 4.25. A sequence $\mathcal{S}_0 \rightarrow \mathcal{S}_1 \rightarrow \dots \rightarrow \mathcal{S}_n$ is *enabled* iff all \mathcal{S}_i are closed and feasible with respect to Σ , and for each $i = 0, \dots, n - 1$ there is some action α_i and some role r with $do(r, \alpha_i) \in \mathcal{S}_i$ such that the execution of α_i will produce the status set \mathcal{S}_{i+1} and the sequence of actions $\alpha_0, \alpha_1, \dots, \alpha_n$ is permitted by the story space.

Example 4.26. Take the actions and Boolean conditions in Example 4.3 and let Σ be the set of deontic constraints in Example 4.23. Let r denote the role new_customer. Then the following sequence of status sets is enabled:

$$\begin{aligned}\mathcal{S}_0 &= \{do(r, \alpha_1), \mathbf{P} do(r, \alpha_1)\} \cup \{\mathbf{F} do(r, \alpha_i) \mid i \neq 1\} \\ \mathcal{S}_1 &= \{\neg\varphi_1, do(r, \alpha_3), \mathbf{P} do(r, \alpha_3)\} \cup \{\mathbf{F} do(r, \alpha_i) \mid i \neq 3\} \\ \mathcal{S}_2 &= \{\varphi_3, do(r, \alpha_5), \mathbf{P} do(r, \alpha_5)\} \cup \{\mathbf{F} do(r, \alpha_i) \mid i \neq 5\} \\ \mathcal{S}_3 &= \{\varphi_5, do(r, \alpha_6), \mathbf{P} do(r, \alpha_6)\} \cup \{\mathbf{F} do(r, \alpha_i) \mid i \neq 6\} \\ \mathcal{S}_4 &= \{\mathbf{P} do(r, \alpha_1)\} \cup \{\mathbf{F} do(r, \alpha_i) \mid i \neq 1\}\end{aligned}$$

The corresponding sequence of actions is $\alpha_1\alpha_3\alpha_5\alpha_6$.

Example 4.26 suggests that not all sequences of actions that are permitted by the plot are compatible with deontic constraints. This raises the question, whether the plot can be rewritten in a way that all stories it defines correspond to an enabled sequence of status sets.

For this we consider only deontic constraints associated with a single role. If Σ is such a set of deontic constraints, we can assume that each formula $\varphi \in \Sigma$ is written as a disjunction of literals, i.e., atoms or negated atoms. If this were not the case, we can transform φ into conjunctive normal form, say $\varphi \equiv \varphi_1 \wedge \dots \wedge \varphi_n$, and replace φ by $\varphi_1, \dots, \varphi_n$. This does not change the semantics, as satisfaction of a conjunction of constraints is equivalent to satisfaction of all conjuncts.

For $\varphi \in \Sigma$ define \mathcal{P}_φ to be the set of Boolean atoms ψ with $\neg\psi$ occurring in φ , negated Boolean atoms $\bar{\psi}$ with ψ occurring in φ , and actions α such that one of $\mathbf{F} do(r, \alpha)$, $\neg\mathbf{P} do(r, \alpha)$, $\neg\mathbf{O} do(r, \alpha)$ or $\neg do(r, \alpha)$ occurs in φ . Let \mathcal{C}_φ be the set of actions β such that $\mathbf{O} do(r, \alpha)$ or $do(r, \alpha)$ occurs in φ . Let $\mathcal{P}_\varphi = \{p_1, \dots, p_k\}$ and $\mathcal{C}_\varphi = \{q_1, \dots, q_m\}$.

Then each permutation σ of $\{1, \dots, k\}$ gives rise to an equation

$$p_{\sigma(1)} \cdots p_{\sigma(k)} = \sum_{j=1}^m p_{\sigma(1)} \cdots p_{\sigma(k)} q_j$$

which degenerates to $p_{\sigma(1)} \cdots p_{\sigma(k)} = 0$ in case $\mathcal{C}_\varphi = \emptyset$.

Example 4.27. If φ is $\varphi_1 \vee \neg\varphi_2 \vee \neg do(r, \alpha)$, then we obtain $\mathcal{P}_\varphi = \{\bar{\varphi}_1, \varphi_2, \alpha\}$, which leads (among others) to the equation $\bar{\varphi}_1\varphi_2\alpha = 0$. In fact, the deontic constraint can be written as $\neg(\neg\varphi_1 \wedge \varphi_2 \wedge do(r, \alpha))$, i.e., it is forbidden to have at the same time $\neg\varphi_1$ and φ_2 and execute α , which is what the equation expresses.

Similarly, for the first constraint in Example 4.23 we obtain $\mathcal{P} = \{\bar{\varphi}_1, \alpha_4\}$ leading to the equation $\bar{\varphi}_1\alpha_4 = 0$, which is a natural consequence: if φ_1 is not satisfied, α_4 is not permitted.

The other two constraints in Example 4.23, however, lead to $\mathcal{P} = \mathcal{C} = \emptyset$, which does not define any equation. Naturally, the constraint expresses a right to execute α_2 and α_4 , which does not press any restrictions on the plot.

Similarly, the constraint $\varphi \equiv \varphi_1 \vee \neg \mathbf{P} \ do(r, \alpha_1) \vee \mathbf{O} \ do(r, \alpha_2)$ leads to $\mathcal{P} = \{\bar{\varphi}_1, \alpha_1\}$ and $\mathcal{C} = \{\alpha_2\}$, which defines the equation $\bar{\varphi}_1 \alpha_1 = \bar{\varphi}_1 \alpha_1 \alpha_2$, i.e., if φ_1 does not hold and α_1 is executed, then also α_2 must be executed.

Thus, each set Σ of deontic constraints defines a set E of equations on the plot. In contrast to the preference rules, however, the equations in E do not just express preferences, but strict restrictions. Technically speaking, if we can rewrite the plot p to a plot p' using the equations in E , then each story enabled by p' , i.e., a sequence of actions, corresponds to a sequence of status sets that is enabled by Σ . We can then apply term rewriting with respect to preference rules to the plot p' .

4.3 Bibliographical Remarks

Kleene algebras (KAs) have been introduced in [421] and extended to Kleene algebras with tests (KATs) in [449, 450, 452]. They have a number of good properties (see e.g., [259, 456]), in particular KATs subsume propositional Hoare logic [321]. KATs have attracted intensive research in theoretical computer science, e.g., [259, 376, 455, 613]. They can be combined with Kleene algebras with relation algebras, e.g., [76]. KATs have been applied for many other application foundations, e.g., for cloud services [532, 534].

Personalisation techniques have already been developed at the beginning of last century within the neurology and psychology research. Later personalisation techniques have been elaborated for marketing (e.g., [492]). They became crucial for website adaptation to the user. With the advent of hypermedia (e.g., [330]), search engines [482], e-commerce and other web applications (e.g., [81, 389, 649]), personalisation issues became a hype within the web research community [596]. The combination with KATs allows however also a formal treatment and computation. Term rewriting techniques [190, 373, 565, 725, 734, 737] and their corresponding properties such as the Church-Rosser property have been applied for computability theory of abstract data types, for system specification, for automatic theorem proving, for the foundations of (functional) programming, etc.

Web information systems use many different kinds of nonmonotonic or non-traditional logics. Deontic logic (e.g., [332, 559, 567, 568, 743] is one of the most prominent.

Key Messages

The **plot** of a storyboard

- describes in an abstract way the possible *navigation paths* through the scenes of a web information system;

- can be formalised by a term in a *Kleene algebra with tests* (KATs);
- can be customised to *preferences* associated with a user type by conditional term rewriting on this KAT.

Deontic constraints associated with a role of an actor

- press *exceptions* on the plot with respect to sequences of enabled status sets;
- define *restrictions* on the plot, for which conditional term rewriting on the associated KAT can be used to obtain a form of the plot that is compatible with the deontic constraints.



Pragmatics of Storyboarding

This chapter lays the pragmatical foundations for the further development of our methodology. Pragmatics is based on the strategic model that we developed in Chapter 2, and gives valuable insight into the development of the storyboard, which we described in Chapter 3. Starting from the branding and mission of a WIS we developed the *utilisation space*, which can be considered as a sketch for content and functionality, the *utilisation portfolio*, which likewise gives a sketch of tasks, actors and goals, and discussed the *utilisation context*. We outlined the value of linguistics by means of word fields and metaphors, and of communication analysis for the development of these sketches.

We first discuss the role of pragmatics in Section 5.1. We then approach the analysis of WIS usage as the first important part of storyboarding pragmatics in Section 5.2. This has been dealt with previously in [735]. For this we provide a deeper analysis of intentions that are associated with the WIS, and classify them using facets for purpose, intent, time and representation, the first of which was already discussed in the strategic model of WISs. These facets of intentions form the basis for the following discussion of *life cases* and *user models*. Life cases have been introduced in [732]. They capture observations of user behaviour in reality, and as such can be used in a pragmatic way to specify scenarios. User models have been investigated in [730]. They capture user and actor profiles, and actor portfolios, from which we obtain a better understanding of the tasks associated with the WIS.

Besides usage analysis storyboard pragmatics addresses WIS portfolios, which have been investigated in [740]. Portfolios address the pragmatics associated with content and functionality. A WIS portfolio consists of two parts: the *information portfolio* and the *story portfolio*. The former one is concerned with information consumed and produced by the WIS users, the latter one captures functionality requirements. WIS portfolios will be discussed in Section 5.3. For this we develop the concept of *persona* as a means to elicit and integrate the information demand of users, and we investigate how the information demand can be combined to define content chunks. In doing so we pay

particularly attention to content chunks for the entry scene. Likewise story portfolios are associated with the detection of functionality chunks based on word fields. However, for these there will be no uniform treatment, so we will have to consider such portfolios separately for various categories of WISs such as e-business, learning, and community WISs, which we will do in Chapter 6.

In Section 5.4 we extend the specification of *contexts* concentrating on the usage and website contexts, and *metaphors*. This was previously addressed in [527] extending work in [89, 401, 399]. The usage context will be classified and specified in more details to show how they impact on life cases, user models and the storyboard. For metaphors we concentrate on their usage in storyboarding.

5.1 The Role of Pragmatics

Pragmatics is part of semiotics, which is concerned with the relationship between signs, semantic concepts and things of reality. This relationship may be pictured by the so-called semiotics triangle. Main branches of semiotics are *syntactics*, which is concerned with the syntax, i.e., the construction of the language, *semantics*, which is concerned with the interpretation of the words of the language, and *pragmatics*, which is concerned with the use of utterances by the user and context of words for the user. Pragmatics permits the use of a variety of semantics depending on the user, the application and the technical environment. Most languages defined in Computer Science have a well-defined syntax; some of them possess a well-defined semantics; few of them use pragmatics through which the meaning might be different for different users.

Syntactics is often based on a constructive or generative approach: Given an alphabet and a set of constructors, the language is defined as the set of expressions that can be generated by the constructors. Constructions may be defined on the basis of grammatical rules.

Semantics of generative languages can be either defined by meta-linguistic semantics, e.g., used for defining the semantics of predicate logics, by procedural or referential semantics, e.g., operational semantics used for defining the semantics of programming languages, or by convention-based semantics used in linguistics. Semantics is often defined on the basis of a set of relational structures that correspond to the signature of the language.

Pragmatics has to be distinguished from pragmatism. Pragmatism means a practical approach to problems or affairs. According to Webster [908] pragmatism is the “balance between principles and practical usage”. Here we are concerned with pragmatics, which is based on the behaviour and demands of users, therefore depends on the understanding of users.

5.1.1 Conceptual Structures in Web Information Systems

The six characteristics introduced in Chapter 1 can be mapped to conceptual structures that are used for storyboard specification:

1. We start with the characteristics used for the strategic layer. Main specification elements used are intention and mission. They are mapped to metaphors, general goals, rhetorical figures, and patterns and grids of web pages discussed later.
2. The scenarios reflect the utilisation by actors, for which we envision a number of stories that correspond to real use. These scenarios may be captured through observation of reality. Story spaces and plots are recorded in various level of detail through the methods discussed in Chapter 3. The stories are reflected in the storyboard.
3. Content specification is the basis for the media types, i.e., data types and their functions, which will be introduced in part III. It combines data specification with user requirements and is reflected in the content portfolio.
4. Functionality is provided by the media types as required by the storyboard. Typical standard functions are navigation, retrieval (search), support functions, and feedback facilities.
5. Context is based on tasks, history, and environment. We use the specification of context for restructuring and functionality enhancement, which will form the basis of XSL transformations and the onion approach that is discussed in the last part of the book.
6. Presentation depends on the intention, the provider, the technical environment available and the users the WIS is targeting at. Presentation results in the layout and the playout of the WIS. *Layout* requires the development of multimedia presentations for each page. *Playout* additionally requires the development of functionality that supports visits of users depending on the story they are currently following to achieve their goals. Layout and playout integrate the chosen metaphors; they depend on chosen page patterns and grids as well as on quality requirements.

Conceptual structures and their association are depicted in [Figure 5.1](#). We may separate the syntactics and pragmatics layers. Arrows are used for representing part-of- or uses- or relates-associations. For instance, the story is based on the user and the functions. Information metaphors relate content to information.

To summarize, the elaboration of storyboard pragmatics in this chapter will focus on users, information and stories plus the supporting metaphors and contexts, as illustrated in the upper part of [Figure 5.1](#). This has already been sketched by the utilisation space, portfolio and context in Chapter 2. These give rise to sketches content, functionality, tasks, actors and goals. For the analysis of pragmatics the investigation of user-specific sample stories will be decisive. These will be called *life cases*.

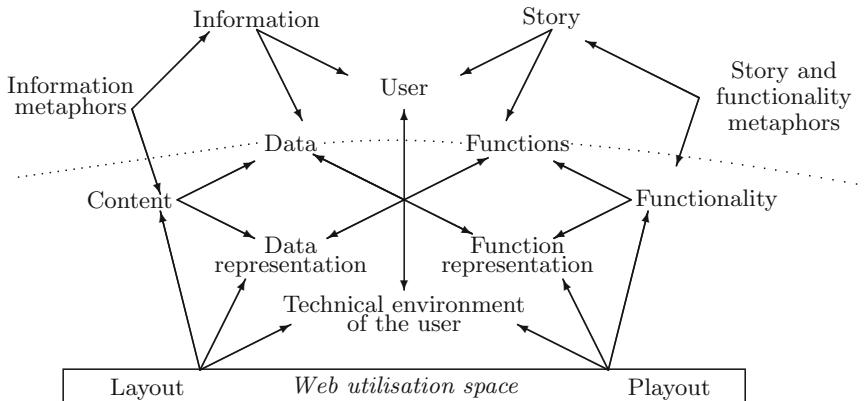


Fig. 5.1. The web utilization space based on the characteristics of WISs

5.1.2 Information Versus Content

We use the notions of information and content in a specific manner. *Information* as processed by humans, is carried by *data* that is perceived or noticed, selected and organized by its receiver, because of his subjective human interests, originating from his instincts, feelings, experience, intuition, common sense, values, beliefs, personal knowledge, or wisdom, simultaneously processed by his cognitive and mental processes, and seamlessly integrated in his recallable knowledge. Content is complex and ready-to-use data. Content management systems are information systems that support extraction, storage and delivery of complex data. Content may be enhanced by *concepts* that specify the semantic meaning of content objects and by *topics* that specify the pragmatic understanding of users.

Therefore, information is directed towards pragmatics, whereas content may be considered to highlight the syntactical dimension. If content is enhanced by concepts and topics, then users are able to capture the meaning and the utilisation of the data they receive. In order to ease perception we use *metaphors*. Metaphors may be separated into those that support perception of information and those that support usage or functionality.

The *information transfer* from a user A to a user B depends on the users A and B , their abilities to send and to receive the data, to observe the data, and to interpret the data. Let us formalise this process. Let s_X denote the function used by a user X for data extraction, transformation, and sending of data. Let r_X denote the corresponding function for data receival and transformation, and let o_X denote the filtering or observation function. The data currently considered by X is denoted by D_X . Finally, data filtered or observed must be interpreted by the user X and integrated into the knowledge K_X a user X has. Let us denote by i_X the binary function from data and knowledge

to knowledge. By default, we extend the function i_X by the time t_{i_X} of the execution of the function.

Thus, the data transfer and information reception (or briefly information transfer) is formally expressed by

$$I_B = i_B(o_B(r_B(s_A(D_A))), K_B, t_{i_B}).$$

In addition, time of sending, receiving, observing, and interpreting can be taken into consideration. In this case we extend the above functions by a time argument. The function s_X is executed at moment t_{s_X} , r_X at t_{r_X} , and o_X at t_{o_X} . We assume $t_{s_A} \leq t_{r_B} \leq t_{o_B} \leq t_{i_B}$ for the time of sending data from A to B . The time of a computation f or data consideration D is denoted by t_f or t_D , respectively. In this extended case the information transfer is formally expressed by

$$I_B = i_B(o_B(r_B(s_A(D_A, t_{s_A}), t_{r_B}), t_{o_B}), K_B, t_{i_B}).$$

The notion of information extends the dimensions of understanding of message displayed in [Figure 2.7](#) to a web communication act that considers senders, receivers, their knowledge and experience. [Figure 5.2](#) displays the multi-layering of communication, the influence of explicit knowledge and experience on the interpretation.

The communication act is specified by

- the communication message with the content or content chunk, the characterisation of the relationship between sender and receiver, the data that are transferred and may lead to information or misinformation, and the presentation,
- the sender, the explicit knowledge the sender may use, and the experience the sender has, and
- the receiver, the explicit knowledge the receiver may use, and the experience the receiver has.

The communication act is the basis for *sagas* that are the basic units of stories.

Users are reflected by actors that are abstractions of groups of users. Pragmatics and syntaxics share data and functions. The functionality is provided through functions and their representations. The web utilisation space depends on the technical environment of the user. It is specified through the layout and the playout. Layout places content on the basis of a data representation and in dependence of the technical environment. Playout is based on functionality and function representations, and depends on the technical environment.

5.2 Usage Analysis

The WIS utilisation space allows to derive sketches for content and functionality, and the utilisation portfolio gives a coarse characterisation of actors, tasks

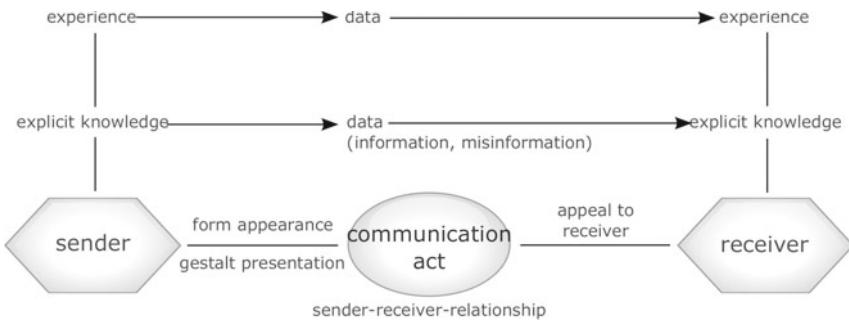


Fig. 5.2. Dimensions of the communication act

and goals. Storyboard pragmatics builds upon this taking a layered approach by first elaborating the intentions and usage, then addressing the development of the main content and functionality, and finally developing supporting contexts and metaphors. These three aspects of pragmatics will be addressed in this and the following two sections. We start with user-specific sample stories, the life cases, the intentions supported by them, and the models of users appearing in them.

5.2.1 Facets of Intention

The description of intention is based on a clear understanding of aims and targets of the WIS including a specification of long-range and short-term targets, expected visitors, characteristics of this audience, tasks performed by the users, necessary and unnecessary content, and finally an understanding of restrictions of usage.

Utilisation scenarios are developed on the basis of intentions, which specify what one intends to accomplish or do, i.e., one has in mind to do or bring about. An intention has four facets that are based on the general characteristics and questions discussed in Chapter 1:

Purpose facet: The *purpose* of stakeholders specifies an anticipated outcome that is intended or that guides the planned actions. It may be based on two additional pieces of information:

- The *aims* specify what is intended to be attained and what is believed to be attainable.
- The *objectives* are more general than the aims. They specify something toward which an effort is directed, e.g., goals or ends of an action.

The purpose is already specified at the strategic layer. It depends on the *audience* intended for the WIS, and is influenced by the *mission* that is determined by the WIS provider.

Intent facet: The *intent* suggests a clearer formulation or greater deliberateness. It distinguishes between the following two aspects:

- The *targets* of stakeholders specify the steps of a plan and its termination or satisfaction conditions.
- The *object* of stakeholders is related to wishes or needs of users, and specifies an effort or activity of a user in order to satisfy the need.

The intent facet is related to *tasks* the user wants to accomplish. The intent may be ordered into *major intents* specifying the main uses of the system and *minor intents* that may be only partially of interest. Intents may be generalized to *themes* that represent classes of intents.

Time facet: The *time* facet is used for specification of the general time restrictions such as

- the *design*, which implies a more carefully calculated plan, and
- the *end*, which stresses the intended effect of an action, often in distinction or contrast to the action or means as such.

Time facets may be very general. In this case, we use *occasions* to represent an entire class of time frames.

Representation facet: Intentions are described or annotated through utterances, words or pictures. The word representation is related to word fields used in the strategic model (see Figure 2.6). The icon representation is based on metaphors. Word fields may be specialised to concept fields discussed later in this chapter. Representation is deeply dependent on the cultural environment and the community that uses the WIS. Intentions can be supported by providing stimuli that rouse or incite activity.

Intentions may be restricted by a *scope*. The scope allows to concentrate on the main aspects. Typical restrictions are the cultural environment, education or other profile properties of potential users, or specific time facets for utilisation of the WIS.

The first two facets of intention have a general form (objective, object) and a more concrete form (aim, target). The purpose facet depends on the mission and the audience, whereas the intent facet depends on the tasks. Therefore, the first facet specifies the ‘what’, the second the ‘how’, and the third facet the ‘when’ of an intention. We may either concentrate on a more general specification or a more concrete one.

The different facets may be considered separately or altogether in a condensed form. The detailed consideration is necessary, whenever a fastidious audience requires sophisticated content. High demands come into consideration due to the content provided, the community that must be satisfied, the sophisticated functionality required for the WIS, or the high attention the WIS gains.

Example 5.1. The purpose facet is often considered to cover ‘soft intentions’ or by ‘business rules’. In an information service a user may be interested or becoming more interested in some content. The intent facet is different and mainly driven by tasks the user tries to solve.

Similarly, we may derive a number of intentions a user has in mind whenever he or she visits an edutainment site:

Aim: An aim of an edutainment user may be to obtain a certificate that proves the success of learning. Aims of providers of edutainment sites might also be binding the learner to some products such as the software of a supporter of the website. Typical general aims within an edutainment site are to grant equal rights to everybody; nobody should have higher rights than other learners.

Objectives: Typical objectives of using an edutainment site are greater ease of learning, greater pleasure or satisfaction during learning, and more fun. Another general objective is security and privacy. Edutainment applications also host confidential information, e.g., information on progress and errors. The learners need to be sure of the security and privacy of their data. They should be informed of the security precautions.

The *audience* of an edutainment site may be rather unspecific or more specific such as students of a college or analysts of a bank.

The *mission* of an edutainment site is to support learning by providing easy-to-grasp knowledge. At the same time, the provider may be interested in increased visitor-to-customer conversion rate, increased number of returning customers, and increased revenue.

The *intent* is related to the more concrete tasks a user has in mind while visiting a WIS.

Example 5.2. Intents come directly from analysing the needs and the demands of users. Some possible *tasks* a user may want to accomplish in an edutainment site are the following:

- A user realises that certain knowledge or information is necessary for solving a task. For instance, an analyst of a bank wants to know what the changes are in the customer community that led to a rise of faulty credits.
- A student in a college needs to know methods for analysing data. In this case, the student is interested in a data mining course that provides material on his or her educational level, permits learning the content interactively, and is comparable to the material the student is currently working with in the college.

Possible intents in an edutainment site are the following:

Objects: There are a number of objects such as faster task completion, successful completion of more tasks, or committing of fewer errors.

Targets: In the bank analyst case, the analyst needs to know data mining methods, to capture the achievements and the disadvantages and pitfalls of such methods.

The *themes* in the bank analyst case could be the interest in learning association methods, preprocessing of data, and prediction methods.

The time facet represents general time restrictions such as the time or the interval for visits of the WIS, the time and the interval of repeated visits, or the temporality that may force a user to leave the site.

Example 5.3. Visitors of a learning site can be characterised by their time dimension, i.e., the time they access and use the site, the access pattern they prefer, and the response time they request from the site.

Design: An edutainment site may allow a visitor to use the material on one shot or to use it step by step. In the latter case, didactics for the elaboration of material becomes an issue of WIS development. Step-by-step learning depends on the cultural environment of the user, the collaboration with other learners, and the interactive facilities of the site.

Furthermore, edutainment sites may be used over a longer time period. Therefore, context-sensitive help becomes important, because some actions or scenes may not be obvious to all users. Making such help available, e.g., through pop-up windows, ensures that the learners do not lose time and interest.

End: Each action of a user leads to some effect that is either made visible to the user, recorded, or changes the state of the learning process itself. The ends of actions must be checked against the intended effect. This kind of consistency check is supported by acceptance conditions used for explicit checkout from scenes.

For edutainment sites, typical *occasions* may be the sudden requirement to explore some knowledge or to prepare a piece of homework. Another occasion might be the sudden demand in quality information due to some news popping up somewhere else.

The representation facet represents the flavour or atmosphere of a site as discussed in Chapter 2. This facet is largely determined by the other three facets.

Example 5.4. An edutainment site that supports bank analysts should strive for an aesthetic and minimalist design. The main target for the representation is to support work. The site needs to ensure a clean and understandable layout. All irrelevant features must be omitted. At the same time an informative feedback is required. The site needs to be consistent throughout. The analysts want to control their way of working. At the same time they become experts of the site and thus need shortcuts and accelerators. Furthermore, they are restricted in their attention to the site, so they need a facility to recover from errors.

An edutainment site oriented to support pupils may follow completely different design principles. Pupils need to be attracted by graphical elements they like, they face in everyday's life, and which make using the site a pleasing experience. Pupils have a less-trained short-term memory. So, the short-term memory load is reduced, if pupils can recognize what they need to know from visible elements of the site.

The set of intentions we should consider may be rather large. Moreover, the faceted representation may become too difficult to manage and to satisfy. The order we may use depends on the audience and on the provider. The audience is oriented towards solving tasks. The provider has a ‘mission’. We can use these two restrictions to figure out which kind of website supports this profile, to harmonize our understanding with the corporate identity, to order the target realisation (short-term, long-term), and finally to develop an understanding how the site will look like in two years from now.

Example 5.5. Aims and targets can be ordered. We can use the mission and goals of the provider and compare them with the tasks the provider wants to support for different audiences. For instance, we may range the priority using a scale from 1 (highest) to 5 (lowest).

aims and targets	community customer	casual customer	reseller	hunter/gatherer	kids	students
selling books	1	1	5	1	4	3
informing on books	1	1	5	1	4	5
reviewing books	1	1	5	1	5	5
selling bestsellers	3	1	4	1	2	5
developing a community	1	5	5	1	4	2

For business or information sites we obtain the following order:

1. primary intentions of the provider;
2. primary necessities and demands of the customers;
3. secondary aims of the provider;
4. stimuli for customers to revisit the system and to buy more;
5. audience education for the customer with our customer community;
6. beauty for customers;
7. customers’ self-realisation.

The specification of the intended audience of the WIS is based on a weighted list of kinds of users, their aims, a comparison of the interaction with users we have so far, and a specification of aims of the audience and reasons for re-visiting.

Example 5.6. Let us consider an information service that supports traveling, e.g., based on the brand

$$(\text{City}, \text{Hotel})^{\text{information}}_2 (\text{Tourist}, \text{Investor})^{\text{inform, book, invest}} .$$

This pattern is based on the following coarse specification of intention:

- *Purpose:* attraction of visitors and building up customer relations based on a *mission* that is centered around area information and attractions.
- *Intent:* travel convenience, hotel room sales and more depending on the *theme* travel support.

- *Occasion*: utilisation of links from the site or other associated sites.
- *Metaphors* used for presentation: survey and collecting basket.

The intention reflects tasks such as travel preparation, supporting visitors of an area, and general information for citizens and tourists.

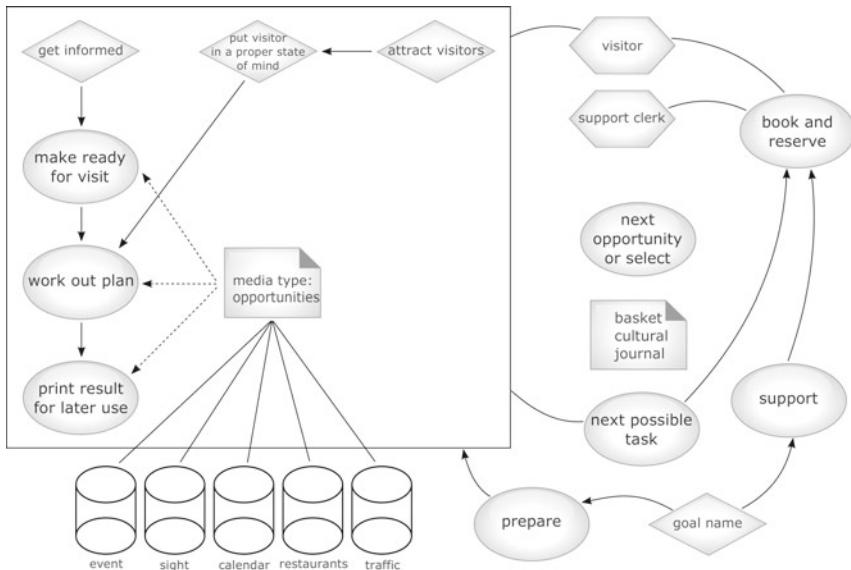


Fig. 5.3. The graphical representation for the intention *prepare for a visit*

Example 5.7. The intention *prepare for a visit* is based on a number of intents such as addressing visitors beforehand for some purpose, use, or activity. It includes the aim to put the visitor of the WIS in a proper state of mind. A task associated with preparation may be to work out the details of a plan in advance. The task may be extended by putting pieces of information together into a compound form or preparing a report. The time facet may range from ‘now’ until ‘next opportunity’. Typical metaphors supporting this intention are baskets, descriptions, and cultural journals.

The intention *prepare for a cinema visit* extends the intention of *prepare for a visit* by a certain content, a refinement of the time facet to the next possible visit, and a refinement of the purpose facet now targeting at entertainment.

We may use a graphical language for the representation of intentions. The language we are using extends the task-goal graphs considered in Chapter 2. Tasks are represented by rectangles. Goals are represented by diamonds. We add now purposes, intents, data types that may be used for task completion, and the time and representation facets. Since purposes and intents refine tasks,

we may place those into the tasks. Purposes, e.g., aims, are represented by rounded rectangles. For objectives we use clouds. Resources, i.e., media types are represented by document icons. Intents are represented by hexagons. In addition, we may add the time facet, the representation facet, and supporting databases. [Figure 5.3](#) displays a graphical presentation of the intention *prepare for a visit*. We have added links that show the dependence among the facets.

We can combine the description of intentions in a semi-formal way as follows. The items in the list are optional except the first one.

Intention space:	\langle intention name \rangle
Purpose:	\langle outcome description \rangle
Aims:	\langle list of aims \rangle
Objectives:	\langle list of objectives \rangle
Intents:	\langle outcome description \rangle
Targets:	\langle list of weighted targets \rangle
Objects:	\langle list of weighted objects \rangle
Themes:	\langle class of intents \rangle
Time:	\langle outcome description \rangle
Design:	\langle general flow \rangle
End:	\langle effects, termination conditions \rangle
Occasion:	\langle list of objectives \rangle
Presentation:	\langle general style guide \rangle
Atmosphere:	\langle general description of atmosphere \rangle
Metaphors:	\langle list of metaphors \rangle
Based On:	\langle tasks, audience, mission, goal \rangle

5.2.2 Life Cases

As already mentioned above, a *life case* is a user-specific story. Life cases build on the tasks and actors we identified in the utilisation portfolio, the discussion of intentions in the previous subsection, and the fragments of functionality in the utilisation space, and thus are decisive for the identification of scenarios of the storyboard.

Life cases allow to overcome the information overload and loss in the hyperspace syndromes typically observed for WISs. For completion of tasks users need the right kind of data at the right moment of time, in the right dose, of the right form, in the complete extent, and within the restrictions agreed upon in advance. Moreover, users are limited in their abilities for verbalisation and digestion of data, and by their habits, practices, and cultural environment.

These limitations may cause intellectual overburdening of users. Most systems that require sophisticated learning courses for their exploration and utilization did not consider these limitations and did not cope with real life situations. The approach we use for avoiding overload is based on observation of real applications before developing the system.

We may extract life cases from observations in reality, which are very useful source, whenever a WIS is going to be developed from scratch or is required to provide a ‘natural’ behaviour. In the latter case users are not required to learn the behaviour of the WIS. Instead, the user can continue using a ‘classical’ behavioural pattern.

Example 5.8. As a motivating example let us consider the life case *relocation* of a person, which consists of

- the change of basic relocation data including the possible removal of data on the old location,
- the change of official documents such as the passport,
- the optional change of relation enhancements such as the registration of pets, relocation of cars,
- the change of personal specific data such as family enhancements, or relationships to religious bodies,
- the change of data for additional relocation announcements such as tax, insurance changes, and
- specific additional tasks such as applications for housing allowances.

The person acts in the role of an *issuer*. We observe that *relocation* is enhanced by the profile of the issuer, by the specific tasks related to the *relocation* of the issuer, by specific laws and regulations, and by advanced functionality required for associating the life case with other life cases.

The life case *relocation* consists of steps such as *change of address data*, *change of data for associated people*, *change of registration data* for cars, pets, etc., *change of specific data*, e.g., data for public authority responsible for aliens, *change of data for social aid*, etc. These steps are bundled together due to their relationship to one person and to one life case. The associations may be represented by adhesion of different steps, e.g., representing the association of steps by a hypergraph, e.g., the one in [Figure 5.4](#).

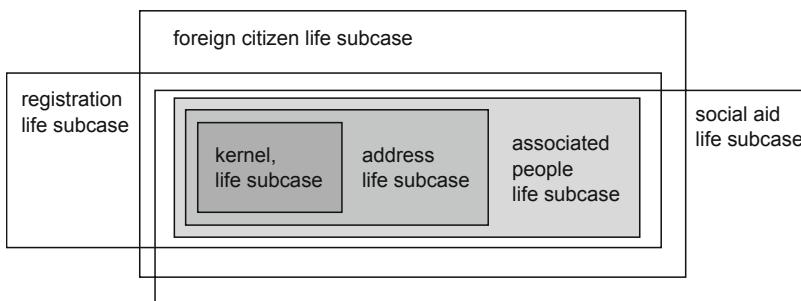


Fig. 5.4. Hierarchically ordered life case steps for *relocation*

The Concept of Life Case

Life cases are characterized by:

Observations: We are interested in the collection and assessment of behaviour relating to the specific application. This would typically involve an observation of behaviour of users in real environments, including a background check that relates usage to intentions, goals or tasks.

Processes: This involves arranging all the actions observed in an application into a main logical and coherent pattern. In some case, deviations of the main pattern must be modelled through exceptions. In most cases, we can use parallel execution of independent actions.

Assessment: This involves the reconstruction of the sequence of actions and specific behaviour of users. This will aid in understanding the role each individual has within the story. It assists in developing the user profile.

Individual profiles: A list of background including physical, and behavioural characteristics of individuals is conducted. This list can also be used for deriving the most appropriate interview technique we discuss below.

Interpersonal coherence: A variation in the activity will relate to variations of other life cases.

Significance of time and place: The choices made also depend on mobility, surroundings, and schedules of events of interest.

Characteristics of the life case: Individuals using a service may be grouped by characteristics. Based on this grouping a similar behaviour is observed.

Experience and skills: Individuals may have their own experience with services provided in real life and thus use different behavioural pattern of service employment.

In general, life case studies are designed to produce a picture of service employment that is as accurate as possible. Determining *why*, *how*, *where*, *when* and *why* a service is called using *what* data provides a better picture for utilisation scenario. As life cases are used for quality control, we must carefully record our observations. We either use a natural language specification or a semi-formal one as described later.

Example 5.9. Let us consider the support of hotel search within an information service. In this case we may observe the behaviour of individuals in travel agencies while seeking hotels. We observe that in most cases search based on associations is preferred over search by main properties such as name, address or facilities. Associations may be of a large variety, e.g., convenience to reach a hotel, location in certain maps, places of interest, or events that have caused the search. Other search criteria may be bargain or bundled offers. At the same time we observe that hotel search is combined with other intentions of users such as visiting cultural institutions. It may depend on results for search of other individuals.

Another example of a life case study is the booking of train tickets depending on individuals, offers of railway companies, circumstances of individuals using trains, etc.

Life Case Development

Life cases may be developed and interpreted step by step:

1. The first step during life case collection is the survey of possible application cases we might consider. The observations could have more than one meaning and may follow a variety of user-related goals. In this case we consider the most likely meaning.
2. The second step involves a deep assessment of the life cases. We extract the different execution orders, the relationship among actions, and the roles individuals play in these actions.
3. The third step extracts those life case characteristics that are distinguishing features and are relevant for time and location. At the same time we search for similarities within these life cases.
4. The final step is concerned with the characterization of potential users, their behavioural patterns, and their roles in various stages of the life case.

Collectively, this information will produce a picture of the life case we are intending to support by a WIS. This may produce further life cases, or may aid in reducing the amount of development. It may result in a prioritisation of life cases under consideration, assist in the linkage of potentially related life cases, assist in assessing the potential of the WIS development, provide the WIS developers with relevant leads and strategies, and keep the WIS development on track and undistracted. These life case are mapped to scenarios in the sequel. The integration of scenarios can also be based on life cases.

Example 5.10. Let us consider an information service for a city or a region, which supports a number of life cases:

- Attracting visitors: The information we are providing is based on some information need visitors may have. The life case follows those that we observe during marketing activities.
- Inhabitants information: The life case is based on selected information chunks that may be of interest to individuals, an ordering of the corresponding available information, and a derivable personal newspaper.
- Informing tourists: The life case is similar to those observed in city information centres.
- Providing official city information to inhabitants: This life case follows the message board metaphor that is used for newspaper information.

During the development of city information services such as the service www.cottbus.de we have made a life case analysis for city information services and detected around 30 different life cases. We can categorize these life cases by the content and information demand of individuals. In this case we distinguish:

- life cases related to tourist content for inhabitants permanently living in the city, for inhabitants temporarily living in the city, for short-term tourists and business people on short term trip, for vacationists planning a trip for more than 5 days, for teenagers with uncertain wishes, for seniors with particular interests, for week-end visitors, for visitors of highlights, festivals, etc.;
- life cases related to official content for inhabitants on search through directory of public authority, for inhabitants on search for emergency cases, for inhabitants orienting themselves before applying at public offices, for inhabitants interested in problems of city management, etc.;
- life cases related to business content, e.g., for investors considering engagement in the area.

For the collection and development of life cases it is normally a good idea to interview the personnel currently providing the service.

Life case should be checked against sufficiency. As they may have a variety of traces, we add two stages to life case analysis:

Exploration: The actual life case is provided to WIS customers and incorporated into their processes. Life case exploration can be conducted in normal environments of the user such as home or work. Then users can show the actions while they are explaining their aims and targets.

Apprehension: Finally, cross checking of life cases and utilisation scenario is used for correction, extension and affirmation of the developed utilisation scenarios.

During life case elicitation and specification users are confronted with sketches of scenarios. We ask what users think about these conceptualisations. The feedback is used for the refinement of life cases. We may use affinity diagramming, in which case we arrange the individual conceptions we discover on a blackboard, associate them, and discuss their relationship to the general characteristics of WISs. Another technique is based on card sorting similar to the model-view-control cards used in software engineering. In this case cards represent simple life situations. Their associations are then used for organising the conceptions. These sketches can also be used for discussing deviations from the normal case and for search of exceptional life cases. Instead of directing users to specific life cases we can show them two or more alternatives for the problem solution.

Life case analysis goes beyond task analysis. It is simple and easy to carry out, lightweight and straightforward to understand, and yet quite accurate in identifying the real demands users of a WIS might have. While observing real life cases and mapping them to life cases that must be supported by the WIS we carry out a user-centered development. Additionally, we may identify and resolve conflicting or competing intentions. These conflicts can be resolved on the basis of life cases by accommodating both intentions, i.e., fulfilling the stakeholder intentions and supporting the demand of users.

Besides observation of real life situations life case detection can also be based on *interview techniques*. Research on artificial intelligence has also resulted in a number of interview techniques:

- Unstructured interviews can be considered as an informal and explorative conversation on goals a user is following and on tasks a user has to complete. Unstructured interviews observe the rules that are applied to brainstorming. All interruptions should be avoided. The questions asked must be short and simple to answer and should be open-ended questions. Interview partners should share their thoughts and experiences. There is no judgement, confrontation or condescension. Users should not be led towards a certain scenario. Unstructured interviews should only be interrupted if clarification is required. They need consecutive feedback in order to give the interviewees the impression that the interviewer is listening. While the interviewees are talking, everything that seems to be important is written down and used for later questions.
- Structured interviews are based on query plans. These plans can be based on the general characteristics of WISs. We start with easy questions on data and main functions and continue with the life cases of their utilization. Context and intentions are more difficult to capture. Structured interviews are also based on features, which can be shown to users for a rating of their importance.
- Life cases can also be detected by observing and critically analysing products of competitors. Elicitation of life cases from existing solutions is based on specifications, results from interviews with business users, excerpts from documents and spreadsheets, analyzed messaging and transactions, knowledge on the solutions that are currently used, meta-data and context information on the utilization within the current framework, and third party information. As reasons for restrictions cannot be captured, the copying of already existing solutions can only be used in exceptional situations.
- Users may also use protocols of “loud thinking”, in which case they are provided with a real-life problem of a kind that they deal with during their working life, and asked to solve it. They imagine that they are solving the problem presented to them. As they do so, they are required to describe each step and the reasons for doing what they do. The transcript of their verbal account is the protocol. In this case, they work and explain their current work. These protocols can be the basis for capturing a scenario. This interview technique should be combined with other interview techniques.
- There are other interview techniques that might be useful for life case elicitation such as the laddering or grid techniques. In these cases, a hierarchical structure of the application domain is formed by asking questions designed to move up, down, and across the hierarchy.

Semi-Formal Representation of Life Cases

Although it is sufficient for life cases to be stated in natural language, we may use a semi-formal representation instead using the following template:

Life case:	\langle life case name \rangle
Characterisation:	\langle outcome description \rangle
Tasks:	\langle list of user tasks \rangle
Problems:	\langle list of problems \rangle
Background:	\langle general characterisation \rangle
Objectives:	\langle list of objectives \rangle
Life case flow:	\langle general graphical description \rangle
Milestones:	\langle graph of milestones \rangle
Content consumed:	\langle consumed content items \rangle
Content produced:	\langle produced content items \rangle
Themes:	\langle class of intents \rangle
Figures:	\langle actors list \rangle
Expected profile:	\langle general profile description \rangle
Collaboration:	\langle general collaboration description \rangle
Context:	\langle general context description \rangle
Time:	\langle temporality limitations \rangle
Place:	\langle assignment of places \rangle
Legacy:	\langle names of documents \rangle
WIS:	\langle general WIS context \rangle
Representation:	\langle general behaviour \rangle
Approaches:	\langle general description of approaches \rangle

This template captures the following components of life cases:

Characterisation: Life cases are characterised on the basis of strategic issues, the problem statement, background and objectives for the life case, the methodology that is used for solving the life case, and by describing the basic modules that are used for solving the life case. The characterisation is harmonized with intentions, tasks and goals.

Life case flow: The life case flow combines the observations we made, the processes involved, and the data which are consumed or produced. The life case flow is mapped to a scenario.

Figures: We develop profiles, especially those of individuals, as part of the WIS utilization portfolio, and interpersonal coherence specifications.

Context: Time and location is explicitly described for life cases. The applicability of life cases is restricted by regulations, laws, and orders. These restrictions are seldom given in an explicit form. In addition, the context of life cases is given by the provider, the intended audience, the utilization history, and the availability of data due to the technical environment. We also use this information for the context specification.

Requirements: Life cases are restricted by habits, general approaches, good practices, and boundary conditions for their application. They presuppose experiences and skills of the users involved.

Note that life cases we intend to support by a WIS can be completely different in real life. Sometimes we need a complete reorganisation of the business activities. In this case we should not map the real life case to a suite of associated scenarios, but rather envision a better organisation of the tasks and goals and then map these to a new envisioned hypothetical life case.

Example 5.11. For illustration let us consider a typical life case that is currently based on mailing and delivery services. Such a life case can be often observed, when a number of institutions is involved in the actual life case and these institutions collaborate loosely using a number of default and exceptional cases. Often, the latter ones are not explicitly stated, but belong to the service context.

For instance, a fee applies to kindergarten or day nurseries. This fee depends on the income of the parent(s) or legal guardians, their social status, the regulations of the day nursery, and the support of the government. Additionally, exemption of fees may be applied for. So, the observed life case consists in the following steps:

- Parents or legal guardians are filling in an application provided by the day nursery. This application is mailed to the youth welfare department.
- The youth welfare department checks the application. They use data provided by other city officials. The result can be the refusal or acceptance of the application, or the request to extend the application by further data.
- In the last case a new application form is sent back to the applicant(s) that will be the basis for calculation of the fees.
- The applicants are providing additional data, e.g., data about their income. This new application form is mailed to the youth welfare department.
- The youth welfare department calculates now the fees that are applicable in the (special) case.
- The decision of the youth welfare department is mailed back to the applicants.
- The applicants may accept the decision made or may file an objection against the decision. The objection is mailed again. It may cause another partial or full assessment.

This life case should not be mapped to a supported life case. Beside unreliability of mailing services a good number of exceptions can be observed. We may use another approach that has been implemented within the Cottbus university service system. Here the applicant becomes the owner of the application and as such all new data associated with the application are made visible to the applicant. Whenever new essential data are added to the application the applicant is notified about it. If a reaction is necessary then

the notification is extended by a deadline tracking life case. The applicant opens a number of views to other actors with rights to see, to update or to extend data. Concurrently, the youth welfare department uses another document tracking system. This system is based on a blackboard approach, i.e., all documents which are currently under consideration are placed onto the blackboard and are ranked by their priority and deadlines. The document tracking system is supported by a work support system. This system contains all necessary regulations, a number of samples, and a data acquisition system that extracts, transforms, and loads data from other collaborating systems. The work support system provides also support for handling exceptional cases.

Thus, the new life case consists of an application delivery and tracking life case, a document handling and assessment life case, and a planning, scheduling and work support life case. These life cases are mapped to scenarios, which are further combined into a single scenario. The three specific life cases become views on the combined one. Furthermore, we observe that the new life case is far more general than the previous one and could be the basis for a generic application processing WIS.

5.2.3 User Models

User modelling has changed the development to human-computer interfaces, and allows to tailor systems to the users, their needs, abilities, and preferences. User modelling is based on the specification of the *user profile* that addresses the characterization of the user, and the specification of the *user portfolio* that describes the user's tasks, involvement, and collaboration on the basis of the mission of the WIS. User models can be extracted from the life cases and intentions on the basis of the utilisation portfolio, which gives already a sketch of actors and their tasks.

In general, user profiles are specified through the *education profile* based on an insight into the knowledge, skills, and abilities of potential users, the *work profile* with a specification of the specific work approaches of users, and the *personality profile* representing the specific properties of users.

Education Profiles

The *education profile* is characterized by properties obtained during education, training, and other educational activities, i.e., education, capabilities, and application knowledge.

The *education* is characterised by the technical and professional training a user got. Technical training emphasises the understanding and practical application of basic principles of science and mathematics. Professional training places major emphasis upon the theories, understanding, and principles in such fields as science, and engineering. It results in erudition, knowledge, and literacy.

Capabilities of the user for task solutions are based on the attainment of proficiency in skills such as understanding the problem area, reasoning capabilities on analogy, realizing variations of the problem solution, solving and handling problems, communication abilities, abilities for explaining results and solutions, and abilities for integration of partial problem solutions.

Application knowledge depends on the application type, the application domain, the application structuring by subjects, the direct structure, the description, the context and environment of the description, the parameters of the application, and application functions:

- The application type covers configuration, production, construction, design, interpretation, error analysis, repair by replacement, rebuilding, and construction, planning, supervision, monitoring, prediction, classification, recognition, instruction, training, consulting, coordination, control, judgement, and evaluation.
- The application domain can be as diverse as science, engineering, e.g., construction, computer engineering, mechanical engineering, telecommunication, energy, environmental, aviation, etc., institutions, government, business, e.g., accounting, administration, finance, production, personnel, procurement, distribution, development, design, material, research, planning, management, etc., military, or journalism.
- Application structuring by the subject refers to title, purpose, domain, evaluation, basic knowledge, and terminology.
- The direct structure captures kinds of associations, e.g., nets, clusters, relations, dimensions, hierarchies, etc., deep or narrow structuring with or without regularity, and internal structuring of the application.
- Application structuring by description addresses the representation with depth, width, level and dimension, solutions with size and complexity, kinds of handling, e.g., procedural, declarative, graphical, combined or mixed, and naming.
- The context of the description deals with associations to the user, applications, space and time, dependence and evaluation by the general context, reasons, purposes, and particularities.
- The environment of the description refers to the dimension of the application in terms of search and solution space.
- Parameters of the application can be time, space and others.
- Application functions are as diverse as accumulate, observe, form, structure, judge, weight, summarize, aggregate, adjust, prove, prefer, determine, estimate, predict, calculate, derive, transfer, control, check, plan, solve, treat, combine, analyse, decide, execute, and feedback.

Work Profiles

The *work profile* is mainly characterized by the task solving knowledge and skills in the application area, i.e., task expertise and experience, system experience, and information and interaction profiles.

The *task expertise* describes the exact and partial knowledge of data, procedures, algorithms, functions, constraints, associations, frames, graphs, schedules, rules, calculi, examples, objects, diagrams, etc. Partial knowledge is characterised by vagueness expressed via uncertainty, incompleteness, inexactness, inconsistency, and heuristics, variants of representations using evidence, evaluation, probability, confirmation, certainty, belief, confidence, support, rough sets, etc., calculi, comments, constraints, etc.

The *task experience* identifies both positive experience, e.g., applicable knowledge, strategies, models and theories, and negative experience, e.g., development, support or knowledge deficits, cooperation and planning support reduction, capability gaps, lack of effectiveness, insufficient competences, etc. In addition, the users' experience in coping with errors, self-organization, planning, and abstraction can be specified.

The *system experience* depends on the systems to be explored and used. It is limited by the user's abilities to cope with information delivered by a system and system-supported collaboration, and to work with systems in parallel to working without system support.

The *information profile* is based on the information needs of a user discussed below, i.e., the intentions in approaching the system, the amount of information a user can cope with, and the complexity of information a user can handle. The information profile can be modelled by database schemata that are extended by a user's preferred way of browsing through information.

The *interaction profile* of a user is determined by his frequency, intensity, and style of utilization of the WIS. Users might be experienced in working with several workplaces or with only one, e.g., a single browser. The interaction profile supports a flexible communication, cooperation, and coordination of a given user.

Personality Profiles

The *personality profile* of a user characterizes his or her *general properties*, his or her preferences in dealing with the WIS, and his or her *polarity profile*, which addresses psychological properties of a user.

General properties of the user or a user group are the *status* in the enterprise, community etc., *formal properties* such as name, address, password, etc., the *context* for performing a task such as position, allocated tasks in development, distribution and application, task area and customer properties, the *psychological and sensory properties* capturing vision, hearing, motoric control, information processing, speed of information uptake, attitude and anxiety, the *background and personality factors* capturing intentions, motivation, expectations, emotions, etc., *training and education*, *behavioural patterns* such as acceptance of system properties and methods, *required guidance* by help and tutoring systems, and *user type*, i.e., whether it refers to a casual user, a knowledgeable user or an expert.

For characterization of *user preferences* we may concentrate on those that are important for WIS development such as preferences for input and output devices and dialogues. For input and output devices this captures the *handling of types* such as text, picture, graphics, audio and video and the required support for these, the *preference of specific forms*, e.g., character-, line-, window- or form-orientation with presentation preferences for colour, shape, size, font, format, contrast, etc., the *required guidance and help during input or output* capturing tutoring, explanations, alternatives, etc., *command preferences*, e.g., the command style, and the *understanding of the input and output tasks* depending on the level and capabilities.

Preferences for the design of dialogues capture *dialogue properties* such as the information load, flexibility, complexity, expressive power, initiated actions, consistency, feedback, etc., *dialogue forms and styles* such as conversation, question/answer form, input/act forms and various presentation styles, *dialogue structuring*, e.g., closed dialogues, open discussions, undirected querying, interrogation, etc., *dialogue control* on amount, quantity, relevance, representation, etc., and *dialogue support*, e.g., by tools.

Polarity Profiles

We may draw special attention to the *polarity profile* that is a part of the personality profile. The polarity profile governs the development of layout and partially also playout. So, we may classify users according to the scale in the following table. Six different categories as by the table below may be used for the characterisation of users. These categories may vary and be further refined using facets.

businesslike	-	lyrical	conventional	-	inventive
unemotional	-	soulful	common	-	extreme
rational	-	sensitive	coated	-	cool
considerate	-	sensual	reliable	-	exceptional
			conventional	-	bohemian
classical	-	modern	traditional	-	vanguard
discrete	-	noisy	old	-	young
ageless	-	modern	colorless	-	gaudy
calm	-	disquietingly	even-tempered	-	exciting
noncommittal	-	intrusive	familiar	-	uninformed
			spiritless	-	lively
tough	-	tender	rustic	-	cosmopolitan
harsh	-	mawkish	natural	-	artificial
geometric	-	bloomy	coltish	-	serious
firm	-	mellow	simple	-	complex
robust	-	delicate	difficult	-	easy
angled	-	round	graceful	-	gracile

These polarity properties may be represented by Kiviat graphs. For instance, [Figure 5.5](#) is commonly known for characterising people from the

Bavaria region in Germany. This self-perception may be used for the development of style-guides or patterns for layout of web pages.

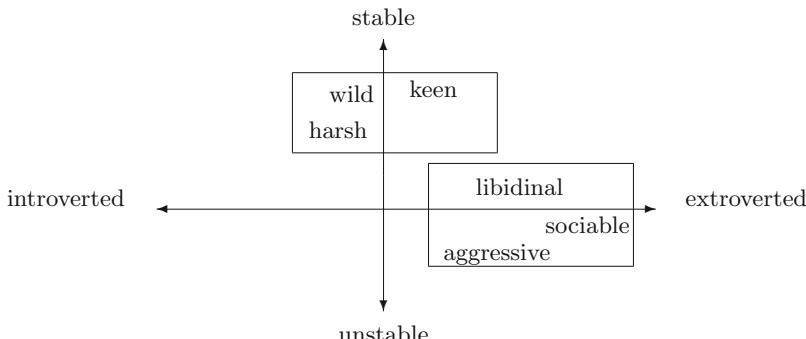


Fig. 5.5. The self-perception of the Bavarian

Representation of User Profiles

User profiles can also be described in a semi-formal way as follows:

User profile:	\langle user profile name \rangle
Education profile:	\langle general description \rangle
Education:	\langle list of degrees \rangle
Capabilities:	\langle list of skills \rangle
Knowledge:	\langle description of knowledge in the application \rangle
Work profile:	\langle general description \rangle
Task expertise:	\langle description of knowledge \rangle
Task experience:	\langle positive and negative experience \rangle
System experience:	\langle experience with infrastructure planned \rangle
Information profile:	\langle information need \rangle
Interaction profile:	\langle interaction properties \rangle
Personality profile:	\langle general description \rangle
General properties:	\langle list of user properties \rangle
Preferences:	\langle list of input/output/dialogue preferences \rangle
Polarity profile:	\langle list of characteristics \rangle
Derivable profiles:	\langle profile description and enforcement \rangle
Security profile:	\langle access control and privacy \rangle
Safety profile:	\langle safety requirements \rangle
Based On:	\langle user goals \rangle
Based On:	\langle user types \rangle

Actor Profiles

We group users by their information demand and requested content, their utilisation patterns, and their specific utilisation and context. The abstraction

of a group of users is called an *actor*. Actors are characterized by profiles and portfolios. Same as for users actor profiles consist of the *education*, *work* and *personality* profiles. Profiles are used for the derivation of preferences. These preferences are used for adaptation of scenes to specific actors. We look at the actor portfolios separately in the next subsection.

Actors may act in the role of an *end-user*, an *antagonist*, or an inactive *non-player character*. The three categories are commonly described by their properties, their abilities, subjects of their actions, and by changes they are imposing.

The profile is also used for the derivation of the *security profile* of the actors. It is specified on the basis of tasks to be performed by the actors, roles they play, and actions that are permitted or forbidden. The security profile includes all aspects of security, i.e.,

- *access control* limiting access only to authorised individuals by applying policies for identification, authorization, and authentication,
- *privacy support* by preventing unauthorized parties from obtaining sensitive information, e.g., through support of anonymity and confidentiality.

This profile may be extended by the *safety profile*.

An actor takes part in some scenes of the storyboard. He or she is characterised by the character, location, actions, intention, and circumstances, i.e., what happened before he or she came there. He must discern the actions he is taking as well as the rhythms of the work as a whole, and he must determine what adjustments must be made in his or her storyboard for each of the other characters. Though actors are abstractions of user groups, we may characterize them by these abstract properties.

Example 5.12. Let us consider an edutainment site. Based on the profile we can derive a number of preferences of learners. The background knowledge leads to different speed and reception of content by the learner. Work abilities and habits influence current work. Learning styles have many many facets depending on the profile of the learner. The social environment with cultural and psychological differences, the support for treatment of history of the learning process without annoying repetitions, and the the content change management with or without refresh are mapped to preferences we apply to the storyboard.

The profiles of actors can be specified using the following template:

Actor profile:	\langle actor profile name \rangle
Grouping criteria:	\langle characteristics of grouping of users \rangle
Information demand:	\langle general description \rangle
Utilisation pattern:	\langle general description \rangle
Specific utilisation:	\langle general description \rangle
Actor context:	\langle general description \rangle

Profiles can also be used for derivation of quality requirements of users.

Example 5.13. For an e-commerce WIS we may derive a number of quality requirements depending on which kind of business the WIS is supporting. For instance, if the system is oriented towards selling after advertisement and presentation then actors that are acting as customers are interested in ease of use, speed, accessibility, previewing, support, community, safety, security, and guidance. If the WIS is supporting configuration sale, e.g., for travel or cars, then customers are interested in customization, feedback, flexibility, previewing, background, contracts, safety, security, privacy, and guidance. If the WIS is supporting direct sale or reselling, e.g., auctions, first-in-first-out sale, or remnant sale then the quality criteria that should be supported are ease of use, accessibility, preview, fun/community, alert, and privacy.

5.2.4 Actor Portfolios

A portfolio is determined by the responsibilities one has and is based on a number of targets. The *actor portfolio* within an application is thus based on

- a set of tasks an actor has or intends to complete and for which solution the actor has the authority and control,
- a description of involvement within the task solution, and
- a collaboration that is necessary for solving the task.

Task modelling means to understand what a user wants to accomplish while visiting the WIS. At the same time, task analysis may lead to a reorganisation of the work processes to be supported. The WIS should support streamlining and augmenting what users do. Task analysis leads to a description of things users do and those they need to know. It does not specify how the task is accomplished. The tasks supported by a WIS need to be representative for the application, important within the application, and completely supported. Task support can be tailored depending on the profile and the context of the actors.

Tasks

Tasks have already been identified as part of the utilisation portfolio. A *task* is a usually assigned piece of work, which often has to be finished within a certain time by a user or a set of users whose duty is its completion. It implies work imposed by a user in authority and obligation or responsibility to perform. A task may consist of subtasks, so we assume that tasks can be constructed on the basis of elementary tasks. Thus, a task is characterized by:

Problem statement: Tasks are associated to problems, for which often a class of solution strategies is provided. Additionally, problems often require collaboration with the local and global systems and with other actors.

Target states: After successfully completing a task we may observe a change in the state. Target states are specified by means of target conditions. Some of the target conditions can be optional. If no optional conditions are given then all conditions are obligatory. Target states correspond to intents.

Initial states: The necessity for tasks enactment is based on the insufficiency of the current state of affairs. Additionally, task enactment conditions may specify under which circumstances we can start task execution.

Profile presupposed for task completion: The completion of a task requires skills, experience and knowledge that must be presupposed by the user, whenever the task is going to be activated. Tasks may be embedded into a certain organizational context. The profile also presupposes a certain technical environment, e.g., communication, information, and workspace systems.

Instruments for task completion: Task enactment is supported by instruments such as actions and data. Problems are solved on the basis of an information demand and within a class of functions that might be used for task solution. Later on the information demand is mapped to database views or media objects. The function utilization is organized on the basis of workflows.

Collaboration profile: A task may be allocated to a single user or a group of collaborating users. In the latter case, the subtasks for each user, the obligations, the time and other restrictions must be given.

Auxiliary conditions: The settling of tasks may be restricted. Typical auxiliary conditions are based on rights for the direct handling and retrieval, roles of the antagonist and the protagonist, and obligations required whenever settling a task.

We extend tasks by a *general task context* that consists of the releasing actor(s), the actors to which task completion and results of completion are reported, the organisational unit that calls for task completion, the activities that are required by users, the aids that can be used for task completion, and the workspace that is supporting task completion. The latter is mapped to session support or to temporary workplaces of users.

Tasks are associated with intents a user or a group of users have. The intents are either objects or targets. In the latter case, we can directly map the task specification to corresponding targets.

Example 5.14. In a learning WIS for data mining learners have very different skills, very different background knowledge, very different approaches to learning, and want to use learning systems whenever this is really necessary. Let us consider tasks to understand the essentials of data mining and seek an algorithm that serves a learner's needs.

1. The learner searches first algorithms that might be applicable within the application under consideration.

2. Next the user needs to understand the algorithm that seems to be the most appropriate. Understanding also includes learning the interpretation of results obtained by applying one of the algorithms to the learner's data. For that, the learner may first look over demonstrations and illustrations for the chosen algorithm.
3. Now the learner experiments with the data. Data are selected and configured for the algorithm. The algorithm is executed and results are obtained.
4. Finally, the results are explained to the learner. He or she may now continue by selecting another method or algorithm and obtain additional insight into the data to be analyzed.

We may map these subtasks to the intents: exploring data mining algorithms, understanding how to prepare data, learning to interpret results obtained.

At the same time this task may be required by a life case [733]. For instance, the learner works as a clerk in a bank and tries to figure out which loans have become faulty and what is the reason for it. In this case, the clerk becomes a learner and tries to learn data mining as much as it is needed for the life case.

Task Execution

The *task execution model* defines what, when, how, by whom, and with which data work has to be done:

- Task activities define how the work is actually done, e.g., by instructing the user or by invoking an application.
- The control flow defines the sequence of activity execution. Depending on the specific WIS several control constructs are available such as decision, alternative, parallelism, loops, start/exit conditions, and different kinds of constraints such as deadlines.
- The data flow specifies how data flows through the task. Depending on the WIS different concepts exist such as input/output-container of activities or local/global variables.

The *result of execution* is the state of affairs reached and the satisfaction of target conditions. Task enactment is supported by pre-determined plans. These plans are scenarios which have a number of stories they are supporting. We may use blackboards for recording open targets. Whenever a target is completed and a task is considered to be successfully fulfilled then we delete this task from the blackboard. The *involvement* of actors within the task solutions is based on the specification of the *role* an actor plays, the *part* the actor plays within the scenario, and the *rights* and *obligations* an actor has within the given role.

The role specifies the behaviour expected of an actor. A role is a comprehensive pattern of behaviour and serves as a strategy for coping with recurrent situations and dealing with the roles of others. A role remains relatively

stable, even though different users occupy the position. A user may have a unique style of role execution, but this is exhibited within the boundaries of the expected behaviour of the actor. Role expectations include both actions and qualities: for instance, in real life a teacher may be expected not only to deliver lectures, assign homework, and prepare examinations but also to be dedicated, concerned, honest, and responsible.

There are two types of roles: declarative and contextual ones. The former ones declare that an actor is playing a particular role, e.g., an actor being identified and as an employee. Contextual roles show how an actor acts within the context of a scenario and how an actor is involved within the context of another scenario. Declarative roles may be modelled by associating the actor to a role type. Contextual roles are modelled by associating an actor with the work effort the actor is assigned to and a role type describing the involvement of the actor. The role type provides a description of the role and can be hierarchically structured. Roles may also be hierarchically structured. At the same time roles may be played in collaboration.

Example 5.15. In an e-business application we can distinguish roles such as worker, customer, and roles within the distribution scenario. The worker role entails roles such as contractor, contact, employee, and sponsor. The customer role takes part in bills to customer, ship to customer, and end-user customer. Within the distribution scenario we may distinguish active and proactive roles, i.e., the distributor and an agent. These roles can again be distinguished into association, competitor, partner, prospect, referrer, regulatory agency, shareholder, supplier, and website visitor.

At the same time several roles are played by people or organizations within the context of project management. A person or organization may be the sponsor, worker, manager, lead, advisor or quality assurance manager for a project.

Users usually occupy several positions, which may or may not be compatible with one another. So actors may play several roles at the same time. In this case, they must have a combined view of the application and a combined access to their capabilities. Therefore, the WIS must provide the ability to *setup* and *manage* these roles, and to *combine* portfolios so that the application maintains to be consistent for each actor.

The *part* associates the atmosphere, mission, objectives, and objects with the involvement of the actor in a scenario. It is refined, when context of the involvement is taken into consideration. The part is based on a categorization of the behavior of users. We may derive that users tackling some problem do not like to wait. The part is described through the kind of interaction actors are preferring for the current tasks and the behaviour of actors we need to support. Therefore, the behavioral stereotype is described through the part.

A role entails certain *obligations* and *rights*. Obligations are obligatory tasks, conduct, service, or functions that arise from the users role under specified conditions. Rights are granted as a peculiar benefit, advantage, or favor

to a role. We specify rights and obligations as conditions that are mapped to conditions on actions an actor can call in a scenario. Obligations and rights include such for functionality, for collaboration, for content, and for dissemination of such to other actors. Obligations and rights may include also the specification of obligations, permissions and prohibitions in the case of exceptional situations.

Collaboration

Actors can collaborate for solving tasks. Such a *collaboration* is specified on the basis of the storyboard, the style of cooperation and the coordination facilities, and the roles of the partners, their responsibilities, their rights, and the protocols they may rely on. Collaboration is separated into *communication*, *coordination*, and *cooperation*. Communication can be based on classical architectures supporting communicating systems. Cooperation is based on an explicit handling of the work processes, the work organization, and on working spaces for collaborating partners. Cooperation can be specified by storyboards. Coordination supports the consistency of work products, of work progress, and is supported by an explicitly specified contract that depends on intentions, on the user's community and on the system supporting collaboration.

Example 5.16. Collaborative learning is a teaching strategy in which small learning groups of learners, each of different levels of ability, use a variety of learning activities to improve their understanding of a subject. Each member of a learning group is responsible not only for learning what is taught, but also for helping group members to learn, thus creating an atmosphere of joint achievement. Learners may apply the knowledge they have obtained during learning steps. Each student may act in different groups in three different roles: the role of an exercise team member, the role of an advisor or expert and the role of an evaluator, who marks and grades the results of the team. The collaborating parties have their own cooperation, communication and coordination space. For instance, the evaluator and the teams share a group communication place and store their results in an answer delivery box. The evaluator delivers assignments to exercise teams through the assignment box.

Actor Portfolios and Life Cases

Portfolios refine the specification of life cases, which are collected by observing real life situations and the data and action flow in them. The life case description allows to deduct the demands for data, functions, performance, and supporting aids such as *workspace* and *workplace support*. The workspace is determined by the portfolio, its support by the profile of the actors involved.

Example 5.17. Let us consider a life case *relocation*. The portfolio for this life case is based on a number of tasks such as: change basic data of issuer, provide

a view on the data to all other interested parties, initiate tasks of associated life cases, archive old data and current changes, handle exceptional tasks, and change data in official documents. The life case is complex due to the large variety of subtasks, the initiation of which depends on the issuer. These tasks are associated with subtasks such as checking whether all necessary documents have been supplied, special handling must be applied, etc.

At the same time we need to consider the involvement of at least four different actors. The issuer raises the change of data task handled by civil servants. If the issuer raises exceptions for data transfer then special support is required by an actor that acts as a data protection official. Finally, the data on the relocation are delivered to a number of actors that play the role of official bodies or support organizations. The security profiles of the issuer and the civil servants require a sophisticated data management, e.g., tax and social security data must be kept in isolation from each other.

The mindmap represented in [Figures 5.6](#) and [5.7](#) summarises a part of the life case relocation, the tasks related to basic changes, to changes of associated parties, to change of data ownership, and to tasks of associated life cases. The actors participating in the life case are restricted to the four main types of actors. Their subtasks are different from the tasks of the issuer.

During analysis of user behaviour a number of principles needs to be observed. These principles form the *portfolio restrictions*.

Non-determinism of user behaviour: We need to cope with all possible options that are plausible. We do not assert that an option for execution will be followed. The various user intents are stored on a blackboard that is used to record tasks and subtasks to be completed.

Intention-based behaviour: We may assume that a user intermittently terminates actions as soon as their aims and their targets have been achieved. A typical implementation support for this principle is realized through the submit button. After submitting data or after hitting the submit button the user starts a new scenario or another scene under the assumption that the current intention has been achieved.

Task-oriented termination: We may assume that a user will terminate the current interaction when their whole task has been achieved. In achieving a target, subsidiary tasks might be generated. Termination is based on termination conditions. For instance, the user of an ATM machine expects that with termination the bank card is returned. Therefore, the user will have the card after termination of the ATM scenario.

Reactive behaviour: Users of a WIS are reacting to stimuli or messages of the WIS the user is observing. Rather than specifying each reaction in a separated form we use generic functions for the specification of this behaviour. Reactive behaviour can be modelled on the basis of observation-reaction pairs.

Collaboration target behaviour: The user enters an interaction with the knowledge of the task dependent on the targets that must be discharged. The

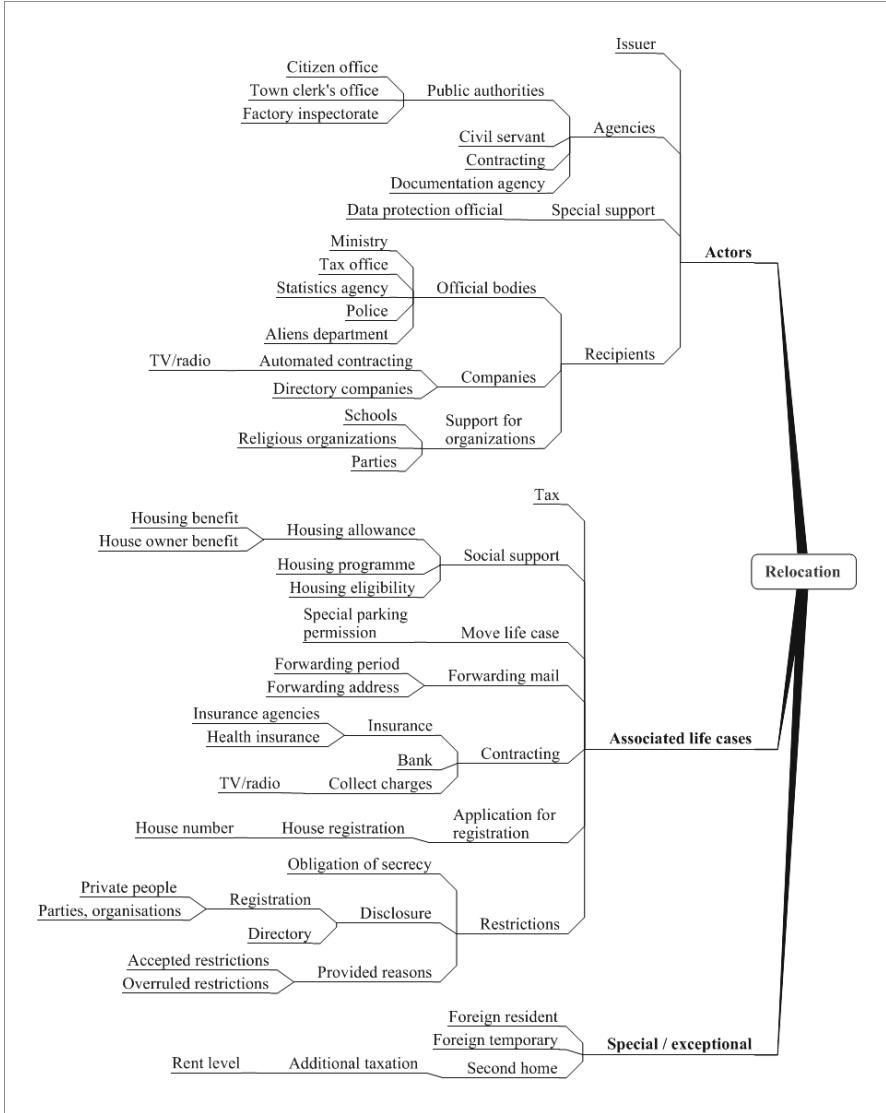


Fig. 5.6. The life case relocation with basic tasks, associated tasks, and actors (left part)

collaboration target is based on a pre-determined plan that is known in advance to the user. Once a target has been achieved the user does not expect to need to repeat the actions again.

Separation of mental and physical actions: Whenever the user commits to taking a physical action, then this action cannot be revoked after a certain

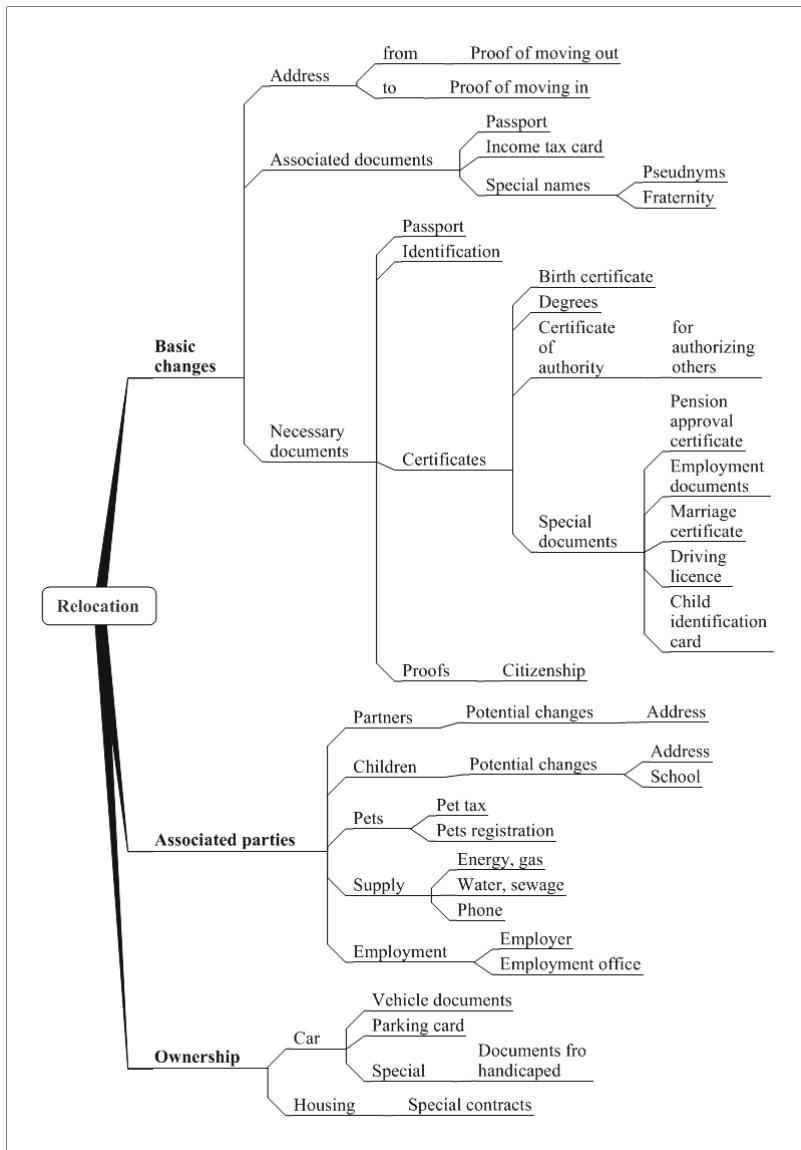


Fig. 5.7. The life case relocation with basic tasks, associated tasks, and actors (right part)

point. Before doing the physical action the user generates the signal sent from the brain to the motoric system. We supply the user model by a guard-action pair linking mental actions with physical actions.

No-option-based completion: A user may abort a scenario when there is no apparent action the user can take that would help to complete the task. If e.g., the user of a railway ticketing system does not find the appropriate option, then the user might give up and might assume that the intentions cannot be achieved. We model this opportunity by a no-option-based addendum that is added to each scene where a user may leave the story.

Relevance: A user chooses an action only, if this action seems to be relevant for achieving the currently considered targets.

These principles can be mapped to *control rules* of the storyboard. For instance, goal-based behaviour is mapped to acceptance conditions of scenes. The intentional non-determinism is reflected in non-deterministic scenarios. Task-oriented termination is mapped to invariants on stories, scenes and scenarios. Reactive behaviour is represented through the transitions between scenes. The user moves with an action from one scene to the next, so the action must be released by the system. We use collaboration targets for the derivation of actions that must be performed by the user. The necessity of actions can be modelled on the basis of deontic logics as described in the previous chapters. They are modelled through action guards. After a target has been achieved it is removed from the task blackboard. The guard-action pairs linking mental and physical actions are also used for refinement of the storyboard. For instance, if a user is not reacting within a certain time frame, then we may assume that the current guard-action is not valid and that the user needs action support or action feedback. The no-option-based addendum is added to each scene as a default addition. We may refine this addendum by actions that a user uses randomly after becoming confused. A possible refinement of the addendum is the incorporation of a wait function. Relevance is a quality criterion we apply during check of each scene. Only relevant actions may be fired by a user.

Portfolios can be specified in a semi-formal way using the template in Table 5.2.4.

Portfolio Context

By explicit specification of the *portfolio context* we have a chance to cope with “errors” a user can make. Typical examples are the unnecessary repetition of an action, jumping back by using the back button of the browser, reversing the order of actions, omitting or delaying actions, replacing one action by another, executing an additional action that is not required or unrelated to the current targets. As it is infeasible to support users in avoiding all these “errors”, we specify the context together with the portfolio.

The context is used for the development of production rules for WIS implementation. A typical list used in some of our projects consists of:

Rules for blackboard control: The scenario is completed whenever the blackboard of tasks becomes empty. The scene data may be logged or removed.

Table 5.1. Template for an actor portfolio

Actor portfolio:	\langle actor portfolio name \rangle
Task:	\langle general description \rangle
Extension of:	\langle General characterisation of tasks \rangle
Characterisation:	\langle general description \rangle
Initial state:	\langle characterisation of the initial state \rangle
Target state:	\langle characterisation of the target state \rangle
Profile:	\langle profile presupposed for solution \rangle
Instruments:	\langle list of instruments for solution \rangle
Collaboration:	\langle specification of collaboration required \rangle
Auxiliary:	\langle list of auxiliary conditions \rangle
Execution:	\langle list of activities, control, data \rangle
Result:	\langle final state, satisfied target conditions \rangle
Actors involvement:	\langle general description \rangle
Role:	\langle description of role \rangle
Part:	\langle behavioural categories and stereotypes \rangle
Collaboration:	\langle general description \rangle
Communication:	\langle protocols and exchange \rangle
Coordination:	\langle contracts and enforcement \rangle
Cooperation:	\langle flow of work \rangle
Restrictions:	\langle general description \rangle
Actor restrictions:	\langle general description \rangle
Environment:	\langle general description \rangle
Based On:	\langle life cases \rangle
Based On:	\langle intentions \rangle
Based On:	\langle general tasks, audience, mission, goal \rangle

Rules for information on actions: Actions that are not initiated by the user but are of interest to him or her need to be reported to the user.

Rules for control flow reporting: Users need a clear indication, if the order of actions that can be initiated by the user is not arbitrary. For example, a common practice is the utilization of a ‘next’ button. If the user does not know what that means then he or she is left in confusion.

Rules for control flow within scenarios: Although a user can choose among several options for actions we need to ensure that options are narrowed down to those that lead to meaningful stories. So, options that are available to the user are reduced at run-time.

Refinement of Life Cases

Profiles and portfolios are used for the refinement of life cases. So far, life cases are specified by a general characterisation, the life case flow, figures, context, and requirements. With the availability of the user model, i.e., the profiles and portfolio of users and actors, we may extend the life case specification by:

Tasks associated with the life case: Life cases are normally not raised in isolation, but are associated with tasks that correspond to problems an actor intends to solve. These tasks have their own dependencies. So, we can associate with a life case other corresponding life cases. This association is mainly used for an extension of the life case under consideration.

Involvement of other actors and collaboration with them: The life case flow is refined by the cooperation specification of all collaborating actors. The involvement includes roles, rights, and obligations. Each actor plays a certain part that is restricted by time, data, processes, and specific restrictions of the portfolio.

Restrictions due to the profiles of participating actors: Due to the profiles of actors life cases are extended in order to support their specific requirements. The personality profiles are used for the generation of requirements according to the atmosphere of the WIS. Security profiles restrict the involvement of actors.

Context specialisation: The general context specified for a life case can be adapted according to the portfolio and the profiles of participating actors.

Example 5.18. The life case *relocation* can be refined using a portfolio that is specific for the state Schleswig-Holstein or for a city such as Cottbus. If there are associated life cases or special cases, then another set of tasks is raised and the issuer is sent to a number of offices. The city information system of Cottbus supports such life cases, e.g., the one depicted in Figures 5.6 and 5.7. In this case, the life case *relocation* is extended by all its associated life cases. In Cottbus one agency is handling all tasks for the issuer. So, the issuer uses the collaboration with the city inhabitants agency. In the background this agency collaborates with all other agencies and recipients.

The life case *relocation* can be raised by any citizen. So, the profiles are rather general. We cannot expect that an issuer has a specific education profile, but we may expect that the issuer has a personality profile. This can be used for the life case flow, e.g., for checking first the existence of documents that are necessary and then raising the basic changes.

This refinement of life cases is reflected in an extension of the life case template:

Life case:	$\langle \text{life case name} \rangle$
Tasks:	$\langle \text{list of tasks associated with the life case} \rangle$
Actors involvement:	$\langle \text{description of actors involvement} \rangle$
Profile restrictions:	$\langle \text{list of restrictions due to actors profile} \rangle$
Context specialisation:	$\langle \text{context embedding} \rangle$

Refinement of life cases permits to derive requirements for the WIS development. These requirements support the designer by providing an input information to sketch the detailed content structure, the interface frames and grids, and the main functionality. Content developers in turn can define the

main content demand needed for the WIS. The information we gain will impact the content the WIS is going to provide, the structuring of this content, the access paths to content, the navigation, the presentation, the user operation, the WIS operation, and the interaction. Furthermore, these life cases may be used as a rich source for usability evaluation. We may also use these refinements for discussing non-functional requirements, which are usually expressed on the basis of user expectations.

The actor profile and the actor portfolio are also used for the derivation of requirements to the WIS.

Workspace: Actors need their own workspace in order to complete their tasks.

This workspace may be provided temporarily, can be partially archived, and can be used for other tasks in the portfolio of the actor. Depending on the involvement of actors who collaborate to solve a task the workspace can also be accessible to other actors.

Session support: Actors are supported by session-defined media objects, i.e., certain content and functionality as defined later in part III. These media objects are generated whenever tasks are started, and may be adapted to the profile of the actors.

Collaboration infrastructure: Collaboration is based on communication, coordination, and cooperation of involved actors. In order to support collaboration an infrastructure is required that is extended or shrunk when new tasks are started or tasks are completed.

Example 5.19. The workspaces for the life case *relocation* depicted in Figures 5.6 and 5.7 are different for the actors. The issuer does not need any workspace. The agency actors have a workspace in which a number of tasks may be bundled. Their workspace can be organized as a blackboard. The actor supporting data protection may use a completely different workspace that is based on monitor techniques. The issuer is supported by session objects that persist until all tasks of the life case are completed. Using this session support the issuer can resume the tasks at any time, e.g., by providing new data. The actors of the agencies collaborate in accordance with the legal restrictions and regulations.

5.3 WIS Portfolios

A WIS portfolio consists of an *information portfolio* and a *story portfolio*. They are mapped to content and functionality specifications, respectively. In doing so we distinguish between *content* provided by the WIS and *information*, which is related to an actor or user.

More precisely, the information portfolio gives rise to *content chunks* that are to be associated with scenes, and the story portfolio gives rise to *functionality chunks*. Content chunks represent content, concepts and topics, and

functionality chunks represent actions. Both together will be mapped to media types, the key concept of the conceptual model, which we will introduce in part III.

To obtain these chunks the word fields associated with life cases are analysed and synthesised through associators of these word fields. The separation into word fields permits the development of story scenes, and the associators reflect the transition from one scene to successor scenes, which can be combined into scenarios. The word fields are then mapped to chunks with chunks associated with nouns representing data and chunks associated with verbs representing functions.

5.3.1 Information Need and Demand

Each WIS user who enters the system with a particular goal has information needs that have to be satisfied by the system. In addition, an active WIS will also request information from its users. We use the term *information consumption* for the information provided by the system to its users, and *information production* for the information entered by a user into the system. Information consumption and information production of an actor for all scenes together define the *information portfolio* of the actor.

When a user enters the WIS, the information needs are usually not known in advance. Part of the needed information may depend on other parts, on decisions made while navigating through the WIS, and even on the information provided by the actor him- or herself. That is, the information consumption and production depends on the path through the WIS, i.e., in our terminology on the story. Therefore, information consumption and production is associated with each scene of the story space.

We distinguish between the *information need* and the *information demand*. The former one refers to a perceived lack of something desirable or useful, while the latter one results from an act of demanding or asking.

The information need is generally related to objectives such as becoming informed. It is based on the intuitive insight that the current information and knowledge is insufficient for the task under consideration, or the necessary information cannot be easily derived from data that is currently available, or the uncertainty, indefiniteness, fuzziness, and contradictions do not permit drawing conclusions.

The information need can be considered to be subjective, but at the same time it can be the reason for a certain user visiting a WIS without an intention that is related to a life case. The information need is based on the conceptual incongruity in which a user's cognitive structure is not adequate to a task, e.g., when a user recognizes that something is wrong in their state of knowledge and desires to resolve the anomaly, when the current state of knowledge is less than what is needed, when internal sense runs out, or when there is insufficient knowledge to cope with gaps, uncertainties or conflicts in a knowledge area. Therefore, as the behaviour of actors is mainly related to life cases and the

portfolio, we have to distinguish between information provided for support of life cases and auxiliary consumed information that is provided to visitors as a service.

The information demand is related to the portfolio under consideration and to the intents. We may distinguish between information that is necessary, desirable, or feasible. The information demand is mapped to the views defining information consumption and production for each scene of the story space as defined above. The information demand is characterised by information that is missing, unknown, necessary for task completion, and directly requested.

We can distinguish between the information demand of an actor and the information demand of a user. As actors represent groups of users, the information demand of a user contains the information demand of an actor. While the information demand of actors is determined by the portfolio, the additional information demand of a user is determined by the user profile.

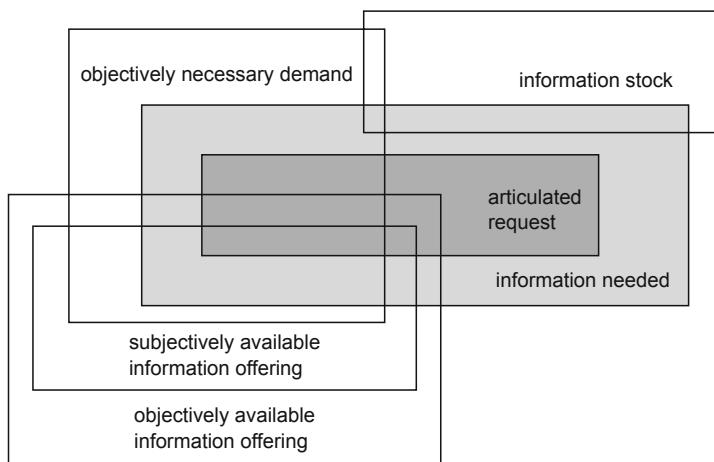


Fig. 5.8. The general relationship between demand, need, request, and availability

Information demand and information need are related to the user and model what is missing. We can also take the *information stock* of users into account, i.e., the information the user has at hand. The information stock relates to the information and educational profiles that are part of the user model. For instance, a user may be an expert for part of the WIS, but a layperson for the rest. Information need, demand, stock and request may not match. The general mismatch is illustrated in [Figure 5.8](#).

5.3.2 The Concept of Persona

The information demand is used to derive the information consumption of each user. This is related to the definition and meaning of information for the

user based on received/requested data, which has to be organized, interpreted, understood, and integrated into his or her knowledge. In general, this would require modelling the user, the specific request of the user, the ability to understand the data, and the skills, which may be infeasible. However, as the information demand of actors is a subset of the one of users represented by the actor, we can use prototypes of individuals called *personae* to determine the information demand. In addition, we model a task-oriented life case of these individuals, and derive the information demand, data requirement, and the specific utilisation requirements.

A *persona* is characterized by an expressive name, profession, intents, technical equipment, behaviour, skills and profile, disabilities, and specific properties such as hobbies and habits. A persona is a typical individual created to describe the typical user based on the life cases, the context, the portfolio, and the profile. User models characterize profiles for education, work, and personality. This characterization can be extended by

- identity with name, pictures (optional), etc.,
- personal characteristics such as age, gender, location, and socio-economic status,
- characterisation of reaction to possible user's error,
- specific observed behaviour including skill sets, behavioural pattern, expertise and background, and
- specific relationships, requirements, and expectations.

Example 5.20. Let us consider an information service of a city and focus on business people. For these we develop the following specific portfolio:

User profile: Jack-of-all-trades is a business man. He intends to visit the city through a short-term visit. He is interested in culture and history. He likes short distances and 4-star or better hotels. He is usually in a hurry. He likes good dining and talking. Additionally, he is familiar with technology.

Intention and information demand: Jack-of-all-trades visits the information site for business trip preparation. His information demand includes hotel information, spare time information for the evenings, and information on the central traffic.

Content requirements: The information demand may be mapped to general city information, information on restaurants, traffic, culture, business clubs, and good dining addresses. Therefore, the information consumption of Jack-of-all-trades must be supported by the corresponding databases. At the same time, Jack-of-all-trades requires a handy booking service.

Life case: The life case we envision includes a brief survey on the city including places to see, the selection of a convenient hotel, a survey of events of interest, the booking procedure for events, a search for good dining places, and some information what else to see and whom to talk to.

Specific utilisation: The collection of data is similar to a basket collection. Jack-of-all-trades prefers shallow navigation and fast search. He is also interested in highlights for the period he is considering.

Jack-of-all-trades can be exemplified to be a specific persona:

- Personal: name: Bernhard Karlowics, age: 48, male, married, lives in northern Germany, profession: business assistant, income: around € 50,000 per year
- Robustness: errors are not taken as own errors, download time is critical
- Kind of user: kind but pretentious, makes quick decisions, interested in history, culture, classical music, usually in a hurry
- Specific behaviour: resumes story also after hours
- Specific interactivity: works alone, with interruptions, no time for concentrated reading, final results expected
- Profile: middle level manager, Masters degree in business and computer engineering, workaholic (around 60 hours per week)
- Portfolio: collection of travel details with confirmation and payment, prefers hotels with four-stars or higher, checking through direct connection to booking services
- Life case: preparing for travel, single use of the system, email confirmation necessary, spare time information, events of interest, dining places, business clubs
- Context: business environment, typically client-server computers at workplace, occasionally mobile phone for contact while on the move

For the persona the following specification template can be used:

Persona:	$\langle \text{persona name} \rangle$
Identity:	$\langle \text{first and last name, photograph} \rangle$
Demography of persona:	$\langle \text{age, gender, location, status} \rangle$
Robustness of WIS usage:	$\langle \text{criticality of errors} \rangle$
Kind of user:	$\langle \text{general description} \rangle$
Specific behaviour:	$\langle \text{general description} \rangle$
Specific interactivity:	$\langle \text{general description} \rangle$
Based On User Profile:	$\langle \text{name} \rangle$
Refined education profile:	$\langle \text{general description} \rangle$
Refined work profile:	$\langle \text{general description} \rangle$
Refined personality profile:	$\langle \text{general description} \rangle$
Based On Portfolio:	$\langle \text{name} \rangle$
Refined task:	$\langle \text{general description} \rangle$
Refined involvement:	$\langle \text{general description} \rangle$
Refined collaboration:	$\langle \text{general description} \rangle$
Refined restrictions:	$\langle \text{general description} \rangle$
Based On Life Case:	$\langle \text{name} \rangle$
Refined characterisation:	$\langle \text{outcome description} \rangle$
Refined life case flow:	$\langle \text{general graphical description} \rangle$
Refined figures:	$\langle \text{actors list} \rangle$
Refined context:	$\langle \text{general context description} \rangle$
Refined representation:	$\langle \text{general behavior} \rangle$
Based On Context:	$\langle \text{name} \rangle$

Refined persona context:	\langle general description \rangle
Refined storyboard context:	\langle general description \rangle
Refined WIS context:	\langle general description \rangle
Refined temporal context:	\langle general description \rangle

The explicit specification of personae has several benefits. They provide communication means within the development team, focus on a specific target set of actors, and help to make assumptions about the target audience. Thus, personae may augment the WIS portfolio specification, but should not be overused.

5.3.3 Content-Centred Analysis

In order to model the information portfolio we collect the information demand of all actors we would like to support. In addition, we can include some specific information demand of users matching with the groups of users. This information demand can be combined into a single *content chunk* that is demanded by all actors of similar steps in the life case. This information demand can later be modelled within a database model.

This combination turns around the viewpoint we have taken so far. We try to envision which content is necessary for which actors or users. We may start with the intention why an actor may demand a given content. The content-centered view allows us to derive a specification of steps in which a certain chunk of content is requested.

Example 5.21. Let us consider a content chunk related to the question why people are visiting a movie display in a cinema. In this case information is sought by visitors due to a number of objectives or targets as shown in [Figure 5.9](#).

A number of actors demand this content chunk for very different reasons and objectives. The content chunk is also related to others, so we can conclude which other content must be given at the same time or in one of the next scenes.

Content-centred analysis is used during brainstorming sessions in which we try to derive scenarios, intentions, and the information need of users. At the same time we can derive the service kind, utilisation, actors, presentation, content, and functions supporting this content. We can also derive directly the intersection among the content chunks, which provides a basis for the development of queries to extract the content demanded from the WIS.

Content chunks are arranged within *content-extended scenarios*, in which each scene is associated with a content chunk. This content chunk combines the consumption of actors, auxiliary content that is provided for the support of users, and the content that is additionally provided due to the intentions of the WIS provider and the context of the current scene. We further enhance

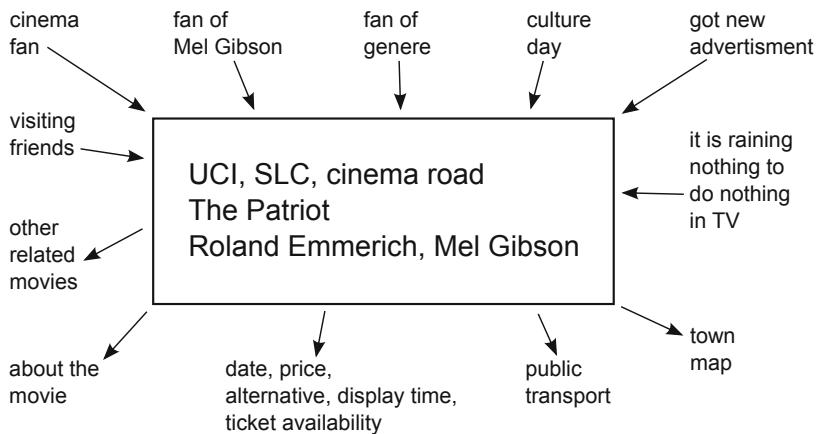


Fig. 5.9. The content chunks requested for the life case of visiting a cinema

this scenario by data that is produced by the actor or that stems from the environment.

Example 5.22. Let us continue Example 5.20 and consider the booking scene. If the actor has decided to choose a particular hotel, then we may associate with the choice action the selection of an identification for the hotel, which extends the booking scene with the content that provides information to the actor on hotels. With the choice of a hotel the actor leaves this scene and produces the selection data. In the next scene the actor can be asked about the payment.

The same scene can also be used in a different scenario, in which the actor first collects all choices in a basket and later confirms those choices, which are the most appropriate.

The information portfolio is also enhanced by information that depends on the WIS context. This can define content associated with control, which normally is internal to the WIS and not accessible by applications, content that is used to determine the transitions and to control the WIS context, e.g., for the pre- and post-scene conditions, transition conditions depending on the databases used, or assignment for collaborating actors, etc., content that might be useful as reference, e.g., meta-information on time, responsibility, and links to other WISs, or application-specific data that is not accessible by the WIS.

The content set required for each step in a life case may become too large. So we must prioritise the development of content chunks. There are two perspectives for prioritising:

- In the usability perspective we evaluate the impact the content has for the the WIS portfolio. The impact is based on the occurrence the content has

in steps and activities and on the number of actors requiring this content. The impact is high if the majority of users need it frequently or cannot continue their work if the content is missing.

- In the economic perspective we evaluate the cost/benefit relation. Since projects have limited budget, contract commitments, resource restrictions, technological constraints, marketplace pressures, and deadlines we must limit the efforts. We may classify those content chunks which have high impact into strategic, high value, targeted and luxuries.

Example 5.23. The www.cottbus.de WIS provides information on hotels, booking services, etc. For the decision which information should be provided we may use a classification along usability and economic perspective as in [Figure 5.10](#). We can then assign final decisions to the evaluation such as approved (✓), under review (◊), and rejected (✗).

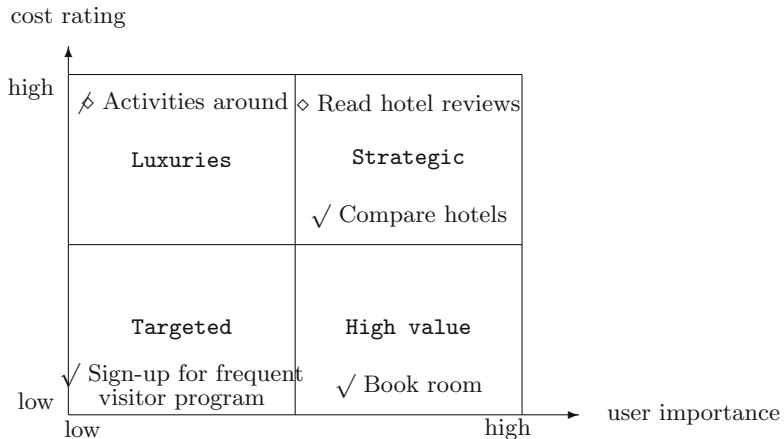


Fig. 5.10. The cost/benefit evaluation of the hotel data in an information service

In a similar way we may evaluate all content chunks we discovered. The evaluation is used for deciding which content is to be developed and to which extent.

Content chunks are often composed only of data, i.e., they relate to media objects on the conceptual model (as we will discuss in part III). The description of the data will give rise to media types, viewpoints, and adaptation facilities. Content is also based on semantics, i.e., associated concepts capturing basic pieces of knowledge or their annotations, and restricted by pre- and postconditions. Content must also be annotated using a commonly agreed vocabulary or a dictionary. This agreement is bound to actors or communities. We denote items of dictionaries by topics. The pragmatics also reflects the context of potential usage. As content is often used in a form that combines content chunks with other content chunks we also specify related content

chunks. In summary, we may use the following specification template for content chunks:

Content chunk:	$\langle\text{content chunk name}\rangle$
Description:	$\langle\text{specification}\rangle$
Semantics:	$\langle\text{general description and knowledge}\rangle$
Concepts:	$\langle\text{list of associated concepts}\rangle$
Precondition:	$\langle\text{condition}\rangle$
Postcondition:	$\langle\text{condition}\rangle$
Pragmatics:	$\langle\text{general description and context}\rangle$
Actor:	$\langle\text{list of actors}\rangle$
Topics:	$\langle\text{list of associated topics}\rangle$
Delivery:	$\langle\text{general description of utilization}\rangle$
Container:	$\langle\text{names of associated containers}\rangle$
Additional content:	$\langle\text{list of content chunks}\rangle$
Usage:	$\langle\text{story space annotation}\rangle$
Enabling Intention:	$\langle\text{list of intentions}\rangle$
Supporting Content:	$\langle\text{specification}\rangle$
Supporting Functions:	$\langle\text{specification}\rangle$
Context:	$\langle\text{context space}\rangle$
Cost-benefit rating:	$\langle\text{usability, costs and benefits}\rangle$
Additional content:	$\langle\text{list of content chunks}\rangle$

5.3.4 Content Chunks for the Entry Scene

The entry scene is associated with a specific content chunk that will give rise to the home page of the WIS. Therefore, it must be developed with greatest care. It must contain all essential information of the WIS on the setting, the environment, the characterisation, the main intentions, activities that can be played with, etc. Therefore, we must balance the list of supported tasks in such a way that they become displayable.

The entry scene concerns the initial situation, the emotional environment, and the main intentions a user may have while visiting the WIS. At the same time, the actor must understand what he or she has to expect when entering the WIS. Objectives of the entry page are to introduce the context and essential actions of the WIS, provide information on collaborating actors, show the theme of the WIS, define the kind of scenario to be played, and support an easy match with the information demand of a user or actor.

The attraction of users requires suspense in the opening, as the first impression decides whether a casual user continues within the WIS. Branding, navigation, content and usage must be balanced. Rules for the entry scene must be based on the characteristics of the WIS, i.e., on intentions, especially objectives, aims and the target audience, on specifics of supported life cases such as need in guidance, feedback, explanation, specific content, and functionality, on complexity of content, which directly leads to the need to support

surveyability, ease of use, and decomposition of content chunks, on variety of functions that must already be provided by the entry scene, on adaptation to user and actor profiles, which requires a rough separation of actors according to their education, work, personality or security profiles, on properties of portfolio such as complexity of problems, need in collaboration, and support for workspace and workplace, and on integration of context, and adaptation of content, functionality, and scenario to the context.

These rules permit the derivation of the general atmosphere and the main intention of the entry scene. Typical examples are categories such as “traditional and serious” for business and work pages, and “energetic” or “vital” providing a flavour of progression and innovation.

Example 5.24. The entry page must accommodate the large variety of visitors and their information needs, for which we can identify a number of content items that must be provided:

- Immediately and clearly communicate the purpose to the visitor: Each visitor must be supported by descriptive wording and images that are easy and quick to understand independent from whether the visit is the first or a repeated one. The values of the WIS must be easily detectable. Visitors’ positive impressions depend on the trustworthiness and the values of the WIS.
- Creating an identity of the WIS (branding): Users need to quickly capture whether a WIS holds a valuable promise, whether they can trust it, and what content is offered.
- Attract by content: A visitor judges a WIS within seconds of entering it. Therefore, content must be attractive, well-organised, easy to browse, and summarized.
- Personalization of content: The WIS should be tailored to the portfolio and profiles of their visitors. In this case, the system does not require users to learn and memorize its facilities.
- Provide an orientation to the visitor: Navigation must be easy to use and may be based on an exploration metaphor. Users should quickly comprehend how to get around, should be not be forced to guess what can be done next.
- Balanced content and functionality: The trade-off between space used for content or functionality can be resolved by developing patterns that support fast detection of items the user is seeking, by focusing on the tasks of the user, and by branding in a constricted form.
- Establishing a cohesive and logical layout: The most important information chunks must be immediately identifiable and located. For this the reading and recognition culture of the users must be taken into account.

The pages of a WIS, especially the home page, are often designed by graphics experts who tend to use a large number of multimedia features. However, layout and playout have to be in accordance with the rules developed

above, which leads to the request to apply screenography, which we will discuss in part IV.

Finally, the entry scene must be considered as one of the main advertisement instruments of the WIS. For this reason, a clear statement of the values of the WIS is the basis for deriving content that conveys these values with the users' first impressions. Typical value properties are

- reliability, availability, actuality, speed, responsiveness, alternatives, ease of use, managing complexity, (filtering) levels, completeness, links, flexibility, support, export, privacy, and guidance for information services, or
- user adaptation, learning styles and preferences, simplicity, support, flexibility, guidance, accessibility, consistency, motivating, clear goals, and responsiveness, or
- portability, speed, shared resources, previewing, alertness, awareness, versioning, mail management, multi-threading, report generation, feedback, mentoring, distribution, flexibility, security, and safety for community sites.

These values of the entry page are a part of the brand of the WIS and must match with the aims and objects that have already been specified for the entire WIS while capturing intentions. We can now combine these requirements for an abstract description of the entry scene using the following template:

Entry scene:	<entry scene name>
Entry scene value:	<main properties>
Intentions:	<list of main supported intentions>
Content:	<list of main content, headlines and blurbs>
Functions:	<list of main functions incl. representation>
Shortcuts:	<anchored list of shortcuts to sub-scenario>
Links:	<content or function anchored list of links>
General navigation:	<content or function anchored list of links>
Actor:	<list of actors, rights, roles>
Story space:	<general description of the story space>
Presentation :	<main properties of the presentation>
Presentation style:	<general description>
Presentation expression:	<scenography and choreography>
Performance:	<maximal time and sequence of download>
Context:	<derived specifics of the context space>

5.3.5 Story Portfolios

The second constituent of the WIS portfolio is the *story portfolio*, which defines a collection of requirements for functionality and WIS utilisation in general. The story portfolio is based on the fragments of functionality as identified in the utilisation space and the life cases, which give rise to a list of

actions as needed by individual users. We abstract from the individual user to extract general function requests, and group these together to *functionality chunks* that can be associated with scenes of the storyboard. These functions are complemented by generic functions for navigation, input, search and retrieval, and selection to give a complete picture of functionality requirements that allows us to derive scenarios of the storyboard.

With respect to navigation we first identify a global navigation structure, which aims at helping users to keep orientation within the WIS. This combines hierarchical navigation via simple backtracking, vertical and horizontal navigation to the next scene and neighbouring scenes, respectively, and ad-hoc navigation via meaningful references as part of the content. In connection with global navigation we identify navigation aids such as explicit maps (floor plan, structuring schema) of the WIS as long as these permit a two-dimensional representation, content description by indexes and catalogues, and markups that may lead to headers, meaningful anchors and icons, keywords, etc. The global navigation structure then has to be refined to capture the identified functions, which are either aligned with global navigation or otherwise give rise to internal, local navigation.

Search and retrieval functions are intrinsically coupled with the consumption of information, and input functions are likewise coupled with information production. The analysis of these functions gives rise to the decision on the specification of subscenarios or the embedding of sophisticated search facilities and navigation through search results. Selection functions give rise to branching in the story space, but the choice may require additional subscenarios.

The story portfolio cannot be described in general and full detail for all different categories of WIS such as e-business, learning and edutainment, communities, etc. However, even though these categories are mixed in real applications, the separate discussion of these categories eases the description of the WIS portfolio. In Chapter 6 we will describe the development of storyboard-ing for the various categories of WISs as outlined in Chapter 1. In Chapter 7 we pick up the discussion of generic functionality.

5.4 Contexts and Metaphors

In Chapter 2 we already introduced the utilisation context and metaphors for the general description of WISs. In this section we further elaborate on these two aspects.

With respect to contexts the *general context* has already been discussed in Section 2. Here we concentrate on the *usage context* and the *website context*. The former one will be refined with respect to actors, storyboard, system, and temporal aspects. For the latter one we analyse provider, developer, organisational and social context.

As to metaphors, their development does not only consist in development of icons and other multimedia elements but can be used to give the WIS a

unique flavour and to prepare for integrity of layout and playout. For this reason, we use metaphors for development of the flavour and the atmosphere of the website.

The illustrated metaphorical structures (metaphor, allegory, metonymy, synecdoches) will now be analysed in order to find common characteristics that can be exploited for site design. We will group and sort the characteristics to get an abstract view at the structural metaphors. The starting point for that will be the general definition of metaphorical structures [130].

5.4.1 Contexts of Web Information Systems

Taking the commonly accepted meaning a context characterises the situation in which a user finds him- or herself at a certain time in a particular location. In this sense context is usually defined only statically referring to the content of a database. Only very few attempts have been made so far to consider context of scenarios or stories.

More generally, we consider context as everything that surrounds a utilisation situation of a WIS by a user and can throw light on its meaning. Therefore, context is characterised by interrelated conditions for the existence and occurrence of the utilisation situation such as the external environment, the internal state, location, time, history, etc. For WISs we need to handle the mental context that is based on the profile of the actor or user, the storyboard context that is based on the story leading to a situation, the data context that is based on the available data, the stakeholder context, and the collaboration context. These different kinds of contexts have an influence on the development of the storyboard and must thus be considered for the development of the WIS.

Example 5.25. Let us consider a travel information system. It is often desirable to resolve the context of utterances. While booking an airline ticket to London the user may be asked for the airport code, for which he or she has a choice between LGW (London Gatwick), LHR (London Heathrow), LON (all London airports in UK), STN (London Stansted), and YXU (London, Ontario, Canada). The context of the travel request can be used to exclude the last option. The context of the airline used so far can be used to exclude two of the others. This context injection is based on the story environment and on content data.

Context Space

When determining context we already know the major life cases we would like to support, the intentions associated with the WIS, the user and actor characterisation on the basis of profiles and portfolios, and the technical environment we are going to use. These restrictions enable a more refined understanding of context within a WIS.

In Chapter 1 we characterised a WIS by six intertwined dimensions, one of which is context. We now relate context to the other dimensions, i.e., to the intentions, the usage, the content, the functionality, and the presentation. As presentation resides on a lower level of abstraction, it does not have an impact on context. Content and functionality will be used for context refinement, which we address later. So, we first concentrate on intention and usage. The user model, the specified set of life cases, and the intention can be used for a disambiguation of the meaning and an injection of context. In doing so we distinguish the following facets of context:

Actor context: The WIS is used by actors for a number of tasks in a variety of involvements and well understood collaboration. These actors impose their quality requirements on the WIS usage as described by their security and privacy profiles. They need additional auxiliary data and auxiliary functions. The variability of use is restricted by the actor's context, which covers the actor's specific tasks and specific data and function demand, and by chosen involvement, while the profile of actors imposes exceptions. The involvement and collaboration of actors is based on assumptions of social behaviour and restrictions due to organisational decisions. These assumptions and restrictions are components of the actor's context.

Storyboard context: The meaning of content and functionality to users depends on the stories, which are based on scenarios that reflect life cases and the portfolios of users or actors. According to the profile of these users a number of quality requirements such as privacy, security and availability must be satisfied. The actor's scenario context describes what the actor needs to understand in order to efficiently and effectively solve his or her tasks in the actual portfolio. The actor determines the policy for following particular stories.

System context: The WIS is developed to support a number of intentions. The purposes and intents lead to a number of decisions on the WIS architecture, the technical environment, and the implementation. The WIS architecture has an impact on its utilisation, which often is only implicit and thus leads to not understandable systems behaviour. The technical environment restricts the user due to restrictions imposed by server, channel and client properties. Adaptation to the current environment is defined as context adaptation to the current channel, to the client infrastructure and to the server load. At the same time a number of legal decisions based on regulations, laws and business rules have been incorporated into the WIS.

Temporal context: The utilisation of a scene by an actor depends on his or her history of utilisation. Actors may interrupt and resume their activities at any moment of time. As they may not be interested in repeating all previous actions they have already successfully completed, the temporal context must be taken into account. Due to availability of content and

functionality the current utilisation may lead to a different story within the same scenario.

We will discuss these various facets of context in more detail later in this section. This entire information forms the *context space*, which brings together the storyboard specification and the contextual information. Typical questions that are answered on the basis of the context space are:

- What content is required by the context space?
- What functionality is required by the context space?
- What has to be changed for the life cases, the storyboard, etc., if context is considered?

As outlined above the context space is determined by the actors, the scenarios, the WIS itself, and the time. It leads to a specialisation of the content, structuring and functionality of the scenes. We will discuss this specialisation in the last part of this chapter.

Context is associated with *desirable properties* of the WIS such as quality criteria and security and privacy requirements. Quality criteria such as suitability for the users or learnability provide obligations for the WIS development process. Though these criteria are rather fuzzy, they lead directly to a number of implementation obligations that must be fulfilled at later stages, i.e., within the development on the implementation layer.

For instance, learnability means comprehensibility, i.e., the WIS must be easy to use, remember, capture and forecast. This requires clarity of the visual representation, predictability, directness and intuitiveness. These properties allow the user to concentrate on the tasks. The workflows and the discourse structure correspond to the expectations of the users and do not lead to surprising situations. They can be based on metaphors and motives taken from the application domain. In the same way other quality criteria can also be mapped to development obligations.

Other properties that may be associated with context refer to the potential utilisation for other tasks outside the scope of the storyboard. In this case we do not integrate the additional tasks into the storyboard, but instead support these tasks, if this is in accordance with our intentions. For instance, we might expect further visits targeting at core concerns of the WIS.

Example 5.26. Sometimes customers may want to use a WIS for a purpose that does not meet the system's mission statement. For example, a customer may use a banking WIS to learn about the loan business, or a bookshop WIS to learn English. Clearly, the larger the gap between the actual customer's intention and the system's mission statement is, the higher the expected costs will be for supporting such customers. If it can be expected that some customers will interact with the WIS in a 'non-standard' way, a decision has to be made whether to support such intentions or not. This implies a modification of the anticipated information space. It shows that our focus on a business model for context modelling is not always a severe restriction.

We may consider three additional context facets:

Provider context: Providers are characterised by their mission, intentions, and specific policies. Additionally, terms of business may be added. Vendors need to understand how to run the WIS economically. Typical parts of this context are intentions of the provider, themes of the website, mission or corporate identity of the site, and occasion and purpose of the visits of actors. Thus, providers may require additional content and functionality due to their mission and policy. They may apply their terms of business and may require a specific layout and playout.

Based on this information, the WIS is extended by provider-specific content and functionality. The storyboard may be altered according to the intentions of the provider, and life cases may be extended or partially supported. Provider-based changes to portfolios are typical for WISs in e-government and e-business applications.

Developer context: The WIS implementation depends on the capability of the developer. Typically we need to take into account the potential environment, e.g., hard- and software, communication channels, the information systems that are to be incorporated, especially the associated databases, and the programming environment developers use.

Organisational and social context: The organisation of task solutions is often already predetermined by the application domain. It follows organisational structures within the institutions involved. We captured a part of these structures already on the basis of the portfolio and modelled it by collaboration. The other parts form the organisational context. Collaboration of partners consists of communication, coordination, and cooperation. Cooperation is based on cooperativity, i.e., the disposition to act in a way that is best helpful for the collaboration partners, taking their intentions, tasks, interests and abilities into account. At the same time, collaboration is established in order to achieve a common goal. Actors choose their actions and organise them such that their chances of success are optimised with respect to the portfolio they are engaged in. Additionally, the social context may be taken into account, which consists of interactive and reactive pressures. Typical social enhancements are socially indicated situations such as welcome greetings, thanking, apologising, and farewell greetings.

Actor Context

Let us next take a deeper look into the facets of the context space, i.e., examining actor, storyboard, system and temporal context in more detail. The context of an actor is based on his or her intentions. According to the actor's profile he or she needs support to fulfil the expectations with respect to the quality of information and work. The social and intellectual interests of the actor may also be part of the actor's context. The actor's profile may be used

for a refinement of the actor's context leading to the following four specific kinds of context:

Actor projection context: Actors may act on their expectations. In this case, they intentionally drop portions of content or functionality and project the current content and functionality to the "normal" case. This projection leads to an implicit context. For instance, within a travel scenario actors are expected to behave like travellers. Another kind of projection is parameter suppression, in which case content or functionality may be dropped or is not noticed whenever it becomes partially irrelevant.

Actor approximation context: Often actors need first a condensed or approximated information that may be refined later. Typical such approximations are attribute value approximations or structural approximations. For instance, the former ones may allow the WIS to provide first an approximate value for the orientation of the user. A common misuse of approximation is pricing by "starting from". Structural approximation permits the use of the same symbol for the original object and an abstraction, hence enables the usage of simpler representations.

Actor ambiguity context: Sometimes the reference of a symbol can be unambiguous within a narrow scope, in which certain limitations apply, but ambiguous in a larger scope without the limitations. A typical unambiguous symbol is the 'next' button in case the next scene lies within the expectations of the actor. Another use of ambiguity can be made by choosing less expressive textual representations. For instance, in a loan application there is no need to clarify that the word 'bank' denotes a financial institution.

Actor mental context: The mental context captures attitudes and knowledge of actors or other kinds of alternative states of affairs such as fiction and user expectations. This context is described in terms of provenance, i.e., relating to real life cases or to expected life cases. Expectations of actors or users can be combined with other more general requirements. The knowledge of the mental context will remain highly incomplete. However, it provides a handle for incorporating users' and actors' expectations.

The actor context is intellectual as well as existential. It contributes to enabling the scenarios and the corresponding stories under consideration. The intellectual part is based on the profile of the actor, on habits, traditions, knowledge, experience, etc. It may be also based on the quality requirements an actor is imposing. The actor context restricts the users that might use the WIS, the way the system will be used, and the portfolios. It is based on the intentions for using the system and the portfolio of the actor, e.g., tasks, involvement, and the collaborations the actor is involved in. The existential part is also related to the portfolio under consideration. It is related to the data and functions currently available or provided, and the technical environment.

The specification of this specific actor context becomes necessary whenever we want to support the work of actors that is close to human communication.

Human communication exploits the context often to an extreme degree, leaving many things implicit. We do not need a complete decontextualisation as long as the actor can interpret the content and functionality that is provided by the WIS. Contexts provide a mechanism by which we can use the simplest presentation, content and functionality, i.e., the ones that makes the fewest distinctions most of the time while transcending to more expressive presentation, content and functionality only when needed.

Due to these contextual abilities we may restrict presentation, content and functionality to those features that are absolutely necessary. These restrictions may result in presentation principles such as sparse utilisation of additional and not directly necessary content or functionality or economy in utilisation of colours, multimedia objects, and texture.

Example 5.27. Let use Example 5.18 for an illustration of this principle. An issuer of the relocation life case expects that his or her personal and identification data are already sufficient for providing him or her all necessary details. So, the context in which the issuer reacts is based on projection and ambiguity context. If we use the information the passport office provides as public information for the city office, then we can adapt the life case directly to the current one. At the same time, the visit of the issuer might be not the first such in his or her life. So, we can now use the information on previous life cases for scaling the life case to the expectations the issuer has.

The adaptation requires some background knowledge on the handling of life cases in other cities, previous visits, and the profile of the issuer. We may then use a number of questions to figure out which further adaption or refinement of the life case is applicable. Since some data on the issuer cannot be stored in the system due to regulations and laws we need to repeatedly obtain these data. So, the data we need to capture within the life case are extended by data we need for figuring out which specific life case is under consideration. At the same time we may use this context information for adapting functionality that is provided.

This specific actor context is combined with the portfolio restrictions. Actors with a non-deterministic behaviour do not use high ambiguity or deep projection. At the same time, their mental context and their approximation context must be rather sophisticated. Actors acting more on intention intensively use all four kinds of actor contexts. Task-oriented and reactive behaviour requires support for mental context. Actors acting in collaborations need additional support for their common disambiguation. If actors do not complete their tasks within one session, they need a well-prepared projection context for the case that they resume their tasks. We shall later map these requirements to adaptation rules and control rules for adaptation.

Storyboard Context

Context has also a storyboard dimension. The actor's context must be combined with the storyboard, life case and portfolio contexts. The latter two selectively condition the situational interest of the actor and the relevance of the current scene for the actor. Based on the relevance we may identify and use properly all the content that should exert the evolution of the current story. We may now use this information for extracting whether a sequence rule, i.e., a rule of the form $s_1 \Rightarrow s_2$ requiring that a visit of scene s_1 should be followed by a visit of scene s_2 , can be applied to the current system usage. The rule may hold in general, but is considered to be not applicable, if the existence of process p_1 leading to scene s_1 does not have a bearing on the existence of process p_2 leading to scene s_2 . Therefore, the incorporation of context and the derivation of relevance has mainly to do with selecting the best story for the user and is thus used for the adaptation of content and functionality.

The storyboard context can be used for deriving the most appropriate content. We aim at delivering the right data to the right actor with the right tools and scope at the right time. As the storyboard context provides a good source for adaptation of content and functionality to the current stage of the scenario we collect context within the storyboard and add this context information to the context space. This context allows a treatment of the expectations of the actor. Therefore, each scene in a scenario is provided with a *pre-scene context*, *scene context*, and a *post-scene context*.

- The pre-scene context consists of all content that has already been delivered to the actor before the appearance at the actual scene. This information can be used to reduce content delivery for scenes. At the same time, this content can be stored in condensed form and made available to the actor when needed, i.e., the actor can revisit the old content whenever this seems to be necessary. Classical browsers only provide a strictly sequential 'back' button for this kind of history management. The pre-context of a scene thus contains all valuable content that is collected during a story, and guarantees the availability of this content when needed.
- The post-scene context consists of a potential playout of scenes that can be entered after the current scene. If an actor needs some information on the next actions, then this context information can be used. This information is valuable for those actors who intend to drop out of the system. It is also a part of the help information. The post-scene context can be enriched by meta-data describing the content that is provided in the next steps or the data to be produced by the actor. In this case, an intelligent interface may forecast the information need for further steps of the storyboard.
- Each scene may also be enriched by superimposed meta-data on the scene, which include everything that could be referenced within the expected consumed and produced data. Typical such references are collaborating actors, retrieval or update data of the current content, and details taken from the

log of the current story. Finally, the scene context may include administrative data such as identification of content currently under consideration. Scene context is enhanced by generic scene information, which can be based on intentions of the WIS. For instance, adverts may be attached to each of the scenes. Default information serves as an exception handling for scenes. If content or functions are currently not available, then default data are provided.

System Context

The system context is determined by the content and the functions that are provided by the web information system. It consists of at least the following four parts:

Source and acquisition: Source and acquisition is an orthogonal dimension of the WIS. A WIS is supported by media objects that belong to media types as we will explore in detail in part III. In a nutshell, a media object is defined by an extended view on some underlying database, which can then serve for provision of content and functionality of an elementary scene. The databases used for the generation of content form the context of the scenario. We may associate with each scenario the subschemata of these databases that are used for generation of consumed data or for integration of data produced by actors in the scenario.

Associated content: The data that are used for consumed and produced information do not exist in isolation. They are usually associated with other data on the basis of integrity constraints or existence constraints, in particular existence constraints are often not explicitly represented as such, but are embedded into the database schemata used. For instance, we usually associate with objects collected in relationship classes those objects collected in the component classes on which they are based. In this case, we assume that objects in component classes of the relationship type co-exist with objects in the relationship class. We need to consider the environment of content that is currently under consideration together with the data that are associated with this content.

Supported functionality: Functions supporting the actions in scenarios are provided by the WIS. These functions have their own control environment. Typical such control mechanisms are logging, concurrency control, and recovery management.

Security: Security concepts describe *encipherment* and *encryption* (keys, authentication, signatures, notarisation, routing control, access control, data integrity, and traffic padding) for data exchange.

Temporal Context

The temporal context appears in a number of variants, e.g., storage time, validity time, display time, user-defined time, transaction time, etc. The tem-

poral context is applicable in a number of combinations. Sometimes it is necessary to use all of them, but often it is often observed that only one variant of this context is necessary.

Versions show the life cycle of the objects under consideration. As scenarios will have their own life cycle we cannot assume that database changes are directly enforced on websites. Moreover, it may be useful to provide the old content as long as an actor continues with the same story. Versions can often be systematically structured by *database system phases*:

- The *initialization phase* permits the development of objects storing initial information. Integrity constraints are applicable in a limited form.
- The *production phase* is the central phase, which consists of runtime querying, modification, transaction management, etc.
- The *maintenance phase* is used in productive database applications for clarification of soft constraints, maintenance of constraints that have been cut out from runtime maintenance, and changing the structuring and functionality of the entire database system. Maintenance phases are used in data warehouse applications for recharging the data warehouse with actual information.
- The *archiving phase* is used for archiving the content of a database in a form that data relevant for historical information can be easily retrieved. No data modification is permitted; the only modification operation is to load new changes to the archive.

Representation of Contexts

We may now combine this context information using the following semi-formal template:

Context:	\langle context name \rangle
Extension of:	\langle General context \rangle
Actor context:	\langle general description \rangle
Projection context:	\langle expectations \rangle
Approximation context:	\langle condensations and abstractions \rangle
Ambiguity context:	\langle scope \rangle
Mental state context:	\langle general description \rangle
Characterisation:	\langle general description \rangle

Storyboard context:	\langle general description \rangle
Pre-scene context:	\langle history of usage \rangle
Post-scene context:	\langle potential continuation \rangle
Scene context:	\langle superimposed meta-data \rangle
WIS context:	\langle general description \rangle
Source and acquisition:	\langle system environment \rangle
Associated content:	\langle content environment \rangle
Supported functionality:	\langle function environment \rangle
Security:	\langle required security functionality \rangle
Temporal context:	\langle general description \rangle
Versioning:	\langle general description \rangle
Development phase:	\langle general description \rangle
Provider context:	\langle general description \rangle
Developer context:	\langle general description \rangle
Organisational and social context:	\langle general description \rangle
Based On:	\langle life cases, portfolio \rangle
Based On:	\langle scenarios \rangle
Based On:	\langle general tasks, audience \rangle
Based On:	\langle mission, goals \rangle

5.4.2 Towards Context Theory

Contexts evolve for actors, scenarios, systems, and over time. We model the relation between different contexts by *lifting relations*. We sketch the basics of this theory in this subsection and discuss it later in detail in Chapter 8.2. Properties that are valid for a certain context may be lifted to another context. This transfer can be based on local model semantics.

Lifting Relations

Recall that a context is determined by actor, storyboard, system and temporal contexts. So let \mathcal{A} denote the set of actors, \mathcal{S} the set of scenarios, \mathcal{W} the set of system characteristics, and \mathcal{T} the set of time units. Then we can take a subset $\mathcal{C} \subseteq \mathcal{A} \times \mathcal{S} \times \mathcal{W} \times \mathcal{T}$ to represent a set of contexts. Furthermore, we use a family of contexts $\{C_i \in \mathcal{C} | i \in I\}$ and a family of statement sets (or theories) $\{\mathbb{T}_i | i \in I\}$ that are associated with these contexts. Of course, the theory \mathbb{T}_i describes the properties of the context C_i .

On these grounds we may use local models $M_{i,j}$ for each of these statement sets assuming that the models we consider are enumerated by the second index. More precisely, the models $M_{i,j}$ determine the meaning of content drawn from a language \mathcal{L} for describing content in view of context C_i . That is, we use a partial mapping $\Psi : \mathcal{L} \times \mathcal{C} \rightarrow \mathcal{M}$, where \mathcal{M} denotes a set of pre-determined meanings for content in \mathcal{L} .

We may now distinguish the formula α occurring in context C_i from the same formula occurring in another context by considering the context index

i , i.e., we consider pairs (α, i) . Lifting relations can be modelled by rules of the form

$$\frac{(\alpha_1, i_1) \dots (\alpha_n, i_n)}{(\alpha, i)} \varphi$$

stating that the formulae $(\alpha_1, i_1) \dots (\alpha_n, i_n)$ can be lifted to (α, i) under the side condition φ . In addition, a compatibility relation among local models is introduced similar to logics that capture possible world semantics. This compatibility relation is used for entailment and satisfiability. This approach allows us to reason locally and then to transfer the knowledge we gained to other contexts.

Based on this coarse clarification of basic notation we develop a number of facilities and extend the specification of the WIS:

Context space: The *content context space* is defined on the basis of the content \mathcal{C} , scenarios \mathcal{S} and actors \mathcal{A} . In Example 5.25 we could use information on the travel and on the airline to exclude options that seem to be less likely. The content context space of a WIS for a given content-meaning pair c, m) consists of precisely those contexts, under which the particular content will have that particular meaning, i.e.,

$$\mathfrak{C}(c, m) = \{(a, s, w, t) \in \mathcal{C} \mid \Psi(c, (a, s, w, t)) = m\}.$$

Adaptation of content, functionality, and scenarios to the context that is currently available is based on *context infusion*. Applying transformation rules we change content, functions, and the presentation. Therefore, we use a context specification for the development of enforcement rules. These rules may restrict scenarios to more specific ones, extend or shrink content, and extend or remove functions.

Life case extension and specialisation: The general life case specification can often be specialised, if context is explicitly injected. We need both the more general life cases and the contextualised ones. Whenever the WIS is revised or extended, we can return to more general life cases and generate another contextualisation. Typical specialisations concern changes in the life case flow. We may specialise the data consumed by an actor in dependence of the actor's context. If we know that actors need special auxiliary information or conversely actors became more knowledgeable during the utilisation of the WIS, then we may adapt the data provided for consumption. At the same time, we can specialise the figures according to the given context. In the same way spatial and temporal information provide a basis for refinement of life cases.

Life cases may be extended to requirements that were collected in the context space. The content context may require a more elaborated content to be provided. The supported functionality may require additional functions, content, or a specific presentation. Intentions may be more specific under consideration of context. For instance, if we want to support

a certain usage of a WIS that was not originally intended but became important in order to maintain frequent visits, then the original life case is extended by those associated life cases.

Development of a context manager: Context is also bound to scenes and thus evolves within a story. We may expect that content enhances context. For this reason, we introduced the pre-scene context. Therefore, a subsystem that manages the context is needed. This context manager uses the lifting rules introduced above for transferring context to context for scenes, collaborating actors, and the WIS as such. The system also supports the rule-based development of logics over time. We cannot require that the rule system is complete, but it must be consistent. A useful property is commutativity, i.e., the results of firing rules do not depend on their order. The context management system enhances the dialogue management system by adapting and specialising the presentation and injecting context into it.

Example 5.28. A typical context extension to functionality is associated with the problem to avoid users getting trapped into losing-track situations. Such situations can be detected based on the user's behaviour, e.g., invoking the help function repeatedly on similar topics, repeatedly positioning on particular locations and performing similar operations on similar data, excessively navigating through information space without invoking any reasonable functionality, looking repeatedly for FAQs on similar topics, attempting to enter a discussion forum, and sending email to the site administrator.

User aid that can be provided for losing-track situations is giving access to a thesaurus of the subsystem the user is accessing. Furthermore, the respective business model may be exposed to the user together with an explanation that is adapted to a particular user type. Similarly, access to a FAQ list suitable for the user and the accessed subsystem may be given. Furthermore, improved search facilities and examples targeting at the subsystem accessed may be provided.

Adaptivity

The idea of adaptivity is to equip the system with enough additional information and rules that would render it possible to engender the right content and functionality for the current situation. That is, the system is supposed to act according to the dictum 'you take care of the specification, and the system will take care of itself and adapt to the current use'.

Two content objects c_1, c_2 are *synonymous* in the context $C_i \in \mathcal{C}$ iff $\psi(c_1, C_i) = \psi(c_2, C_i)$. They are *totally synonymous* iff $\psi(c_1, C_i) = \psi(c_2, C_i)$ holds for all contexts $C_i \in \mathcal{C}$. They are *epistemically synonymous* within a scenario s for an actor a iff $\psi(c_1, C_i) = \psi(c_2, C_i)$ holds for all contexts $C_i \in \mathcal{C}$ associated with a and s .

Applications often require adaptation of processing context, e.g., to

- actual environments such as client, server, and current communication channel,
- user rights, roles, obligations, and prohibitions,
- content required for the portfolio of the current user,
- the actual user with his or her preferences,
- the level of task completion depending on the user, and
- the user's completion history.

Consider for instance e-learning or e-government websites discussed in Chapter 1. Citizens may apply for a primary place of residence. In this case, their passport must be changed; otherwise, no change is required. Citizens with school-age children may have to complete additional documents. Completed documents may be decomposed into a suite of documents due to legal restrictions, e.g., by a data protection act requiring that data for city officials and service offices such as the unemployment agency must be separated.

Depending on the role of users, story completion may be scheduled sequentially for some users or in parallel for others. For instance, clerks in a city office may consider documents in parallel, while citizens complete their documents in a sequential mode.

Example 5.29. Adaptivity may be required at run-time. For instance, people with foreign citizenship may be required to apply for a residence permit. Users may require a varying support depending on the environment that is used for the completion of documents. Users should be supported whenever they are interrupted during task completion.

These requirements lead directly to the requirement to develop a facility for *mutable, adaptable scenarios* for different users, portfolios, and contexts. We shall return to this requirement after introducing templates in the next section. It is our target to develop generic scenarios that can be refined to scenarios by injecting context. This approach is more widely used for WISs than one would expect. For instance, almost all information sites of cities and regions provide a very similar hotel or event search. The reason is not the existence of a development monopoly but rather the evolution of these search facilities to semi-standards. These standards are not officially agreed, but have been formed by copying successful solutions.

5.4.3 The Metaphor Concept for Web Information Systems

The definitions of the metaphorical structures show two main aspects: the (language) representation of a metaphorical structure, and the cognitive understanding, i.e., the characterization of the actions and actors underlying the use of metaphorical language. The intent behind the application of metaphors can be to achieve an unusual effect, use an insufficient expression to describe a thing or action, or make an abstraction of one or more things or actions. Based on the intentions, the usage and the functionality of metaphors can be very different. For this reason, we distinguish between

- internal functions extending the expressive power,
- predicative functions, i.e., modelling and describing by analogy,
- heuristic functions by adding information which can be interpreted differently,
- emotional functions, e.g., using the intuitive experience for complex associations,
- social functions, e.g., developing specific static presentations,
- rhetoric or persuading functions, e.g., changing users behaviour, and
- aesthetic functions to please the user.

Metaphorical structures are realized through their usage context. Their interpretation depends on the common experience of the sender and the receiver. In linguistics, different forms of metaphorical structures are used:

- *Personalized* structures can be used to express similarity. For instance, the expression ‘the sun is laughing’ is using a human property such as laughing for a property of a different object meaning that the sun is shining bright.
- *Allegoric* structures, e.g., ‘friend Hein’ for death are based on a common culture of the receiver and the sender
- *Symbolic* structures have a rather concentrated message.
- *Synoptic* structures assemble words from different areas, e.g., ‘dark tones’.

Metaphorical structures change the meaning of an expression by adding new, extended, shortened, or different meanings. This defines a substitution of meaning with a direction. Therefore, we obtain three orthogonal classification schemata: abstraction referring to specialization and generalization, direction referring to inanimate or animated properties, and interkind or -type transferring or reasoning by analogy.

Starting from the analysis of characteristics and representation forms of metaphorical structures we can construct a dictionary, which is needed to find the right metaphorical structures for WIS design in a certain context.

The intuitive usage of metaphorical structures in everyday life is the reason that users cannot easily give examples of metaphorical structures when needed. On the other hand, systems like MIDAS [375, 187] have shown that an automatic interpretation of metaphors is actually not really possible. The problem is the context-based interpretation.

Therefore, we use a *feature semantic approach* to access the metaphorical structures. In the dictionary, different representation forms are stored with their relation to the known characteristics. Idioms, phrases and dialogues can be metaphorical structures selected from common dictionaries. The potential metaphorical structures are classified using the illustrated properties abstraction, direction, etc. In addition, we may consider common characteristics such as colour that are directly related to the metaphor specification frame.

5.4.4 Application of Metaphors in Storyboarding

Searching a concrete metaphor we define one or some dominant properties and take an applicable representation form.

Metaphorical structures are used for the exchange of semantical units. The exchange is dependent on the receiver, the sender, the ‘tertium comparationis’ (dominant properties), and the context (language context, intentions of the sender and expectations of the receiver). We distinguish a number of use cases:

Metaphorical structures for selecting information. Metaphors can introduce unwanted baggage or unwanted restrictions. For example, users might expect a virtual shopping mall to be staffed by a shop-assistant who will answer questions and who will help in selection. Users visiting a virtual bookshop might expect that somebody will help in selection instead of advising to buy the best advertised products. From the other side, metaphorical structures might introduce additional restrictions. For example, the telephone symbol is often used in web sites for links to addresses, phone, fax, etc.

We often meet internet pages whose graphical items and metaphors are overwhelming. Users cannot concentrate on the content or are not able to capture the mental model of the site. Metaphors can be used in a completely misleading fashion. For instance, one of the design companies is using a pencil as a metaphor. This metaphor is important for the company itself, but not for the visitors of their page or their customers. The message of working thoroughly with appropriate instruments can be of importance. However, it is a side message. Thus, metaphorical structures are wrongly developed if they are carried forward into the site itself. Even worse, they can hamper usability. Typical examples of such metaphors can be found in everyday life. For this reason, they should be applied with care.

Since metaphorical structures are supporting dialogue objects, they do not play a major role in a site. They do not have a dominant role. They support the intention of the site. They are an element of the form and can partially present the content of a page. Metaphorical structures should be integrated into the site development process. The development of sites includes, as mentioned above, different perspectives. Metaphorical structures can introduce inconsistency into the site. Metaphorical structures should be a part of the interface.

Metaphorical structures for complex information. Commonly adopted metaphors are stand-alone metaphors denoting mainly atomic information units such as phone, address, next step, special action. They are displayed by buttons. Metaphorical structures can be more sophisticated. They can represent a complete mission of a page. They can be displayed in very different fashion such as background color, background picture, texture, and special alphabets, etc.

Some of the principles can be applied to metaphor development of sites [927]. The principles are fairly unknown to computer science where

metaphor development has been understood mainly as development of representation of widgets in the GUI community or where metaphors are used to describe areas of research in a simple way (information highway, artificial intelligence, agent technology).

Metaphorical structures for teaching the user. As metaphorical structures are intended to be used several times, they have a learning effect [381]. They are used to give a better description. They are intended to be recognized by the user. They have a specification functionality. They are both common and unfamiliar. They extend the understanding. They force an event of immediate learning through playing. They can contain requests to act.

Composition of metaphors. Metaphors can be combined in order to weaken or to strengthen properties. For instance, the event database engine in Cottbusnet is based on several metaphors at the same time: TV program journal, cultural journal, supermarket basket, phone book, landscape, and internet browser. The combination of these metaphors implies the functionality of query interfaces for daily, weekly and category-based display of information. The presentation of search results can be chronological, alphabetical, categorical or can be based on landscapes. Program journals have a specific survey approach and use escort information. Cultural journals discuss events with background information. Baskets allow interactive combination of objects. Internet browsers have additional support functions such as bookmarks. The combination of these metaphors also suppress properties such as piles in baskets.

Metaphorical structures impact the implementation of a WIS. They can be used to represent a story or scenario, escort information about a scene, specific functionality such as navigation, searching, labeling, etc., a class of content objects, or application restrictions (user profile, communication channel, users platform). Metaphors can be used as singleton elements, parts of the page representation, or as a substantial element of the whole page.

Summarizing the characterization of metaphorical structures presented above and applying this to metaphorical structure in WISs, we obtain the following specification frame for metaphorical structures:

- name of the metaphorical structure,
- property specification relating to application areas, weighing them by an intensity value, and based on the intensity deducting the dominance of properties for a given application area,
- class of the metaphorical structure (personalized, allegoric, symbolic, or synoptic),
- meaning for different groups of actors in different cultural contexts, and
- representation specification relating to representation patterns such as images, colour, words, sentences, etc.

Metaphorical structures can be used for structural or behavioural information, or both. Structural metaphors are used to represent the structure of objects, the underlying types (or their properties) of objects, or the content of objects. Typical examples are dealership sites. People have a mental model of how dealerships are organized. Behavioural metaphors can be used to describe actions, the current state of an object, tools or agents, or actors. They make a connection between the tasks which can be performed in a traditional environment and those the user can perform in the new environment. Shopping baskets, browsing through shelves in a library, or browsing in a book are examples of commonly used behavioural metaphors.

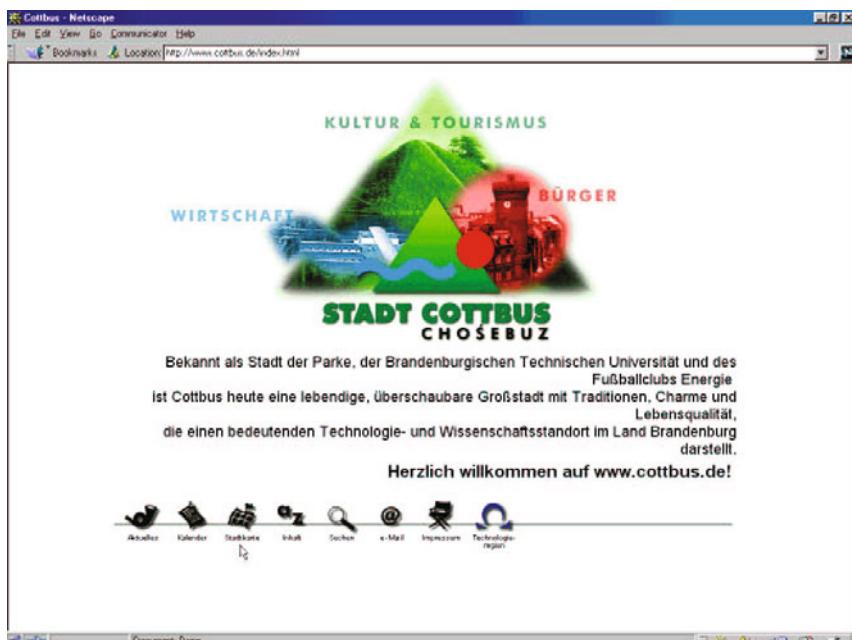


Fig. 5.11. Metaphors used for the Cottbusnet project (www.cottbus.de, second generation)

Visual metaphors are based on familiar images. For instance, yellow and white phone buttons direct to print-based yellow and print-based white pages. They are mainly used to support navigation. There are other visual metaphors beyond labels of icons.

Example 5.30. In Cottbusnet (cf. Figure 5.11), we used colors to denote the subsite, with the colors ‘blue’, ‘red’,¹ and ‘green’ denoting business informa-

¹ The heraldic animal of Brandenburg is the red eagle. All official governmental and community documents use the red color.

tion, inhabitant information and tourist information, respectively. The presentation of historic information, can be based on the metaphor of the ‘old parchment’.

Hierarchical structures can be visualized through different metaphors such as plain two-dimensional representations, cone trees, hyperbolical figures, perspectives in rooms or as abstract information landscapes. The Fürst Pückler pyramid used in *cottbus.de* combines a Cottbus identity picture with a functionality of selecting an appropriate page for the next step. While selecting a page, the actor is classified into being an inhabitant of the town, being a visitor, or being interested in business.

5.5 Bibliographical Remarks

Pragmatics is the study of the relations between languages and their users [842, 693]. It is concerned with the relationship of sentences to the environment in which they occur. It is different from pragmatism as a practical approach to problems and affairs. The conceptual structure of a WIS has been developed in [527, 730, 732, 740, 735]. Layout and playout has been investigated [580, 620] and is the topic of the forthcoming book [582].

The notions of data, information, and knowledge are different. Data are characterised in the syntax dimension. Knowledge is related to semantics. Information can be understood in the pragmatics dimension. We use the anthropomorphic notion [852] instead of the syntactic, semantic, or pragmatistic notions [845]. It relates information to the user and the user demands. Usage includes generally accepted practice or procedure from one side and the mode and manner of using a WIS.

The intention of a user [236] is characterised by four facets representing the why-when-what-how dimensions. They can be enhanced to the rhetorical frame “who says what, when, where, why, in what way, by what means” (Quis, quid, quando, ubi, cur, quem ad modum, quibus adminiculis) that has been introduced by Hermagoras of Temnos or Cicero. Another extension is the W*H frame [175].

Life cases generalise the concept of business user cases [537, 678], lifestreams [219], and life events (e.g., for e-government sites [673]) [123, 548]. They are determined by events in WIS user’s lives that alter their work, role, position, and/or resources.

A user model is defined to represent a collection of personal data associated with a specific user to be used as the basis for adaptive changes to the system’s behaviour [230]. According to [230] there are four categories of user models. In static user models (1), once the user related data is gathered, it is not changed again. Shifts in users’ preferences are not registered and no learning algorithms are used to alter the model. Dynamic user models (2) allow a more up to date representation of users. Changes are noticed and data related to it is updated to take the changed needs and goals of the users into account. Stereotype based

user models (3) are based on demographic statistics. Users are classified into common stereotypes and the system adapts to this stereotype. Highly adaptive (4) user models try to represent one particular user and therefore allow a very high adaptability of the system. In contrast to stereotype based user models they do not rely on demographic statistics but aim to find a specific solution for each user by recognising the different context related factors of the user. Our approach [747, 730] combines this research with the findings in [29, 313, 432], psychology (e.g., [474]), and in AI and HCI research (e.g., [125, 274]).

Portfolio management is based on task description [385, 635, 740, 735, 862], activity theory [498, 636], and organisation theory (e.g., [48]). The task description combines task conceptions and problems used in AI and methodological research [366, 657, 821]. WIS cannot be developed without collaboration with the user community and specific approaches for user adaptation, e.g., in the Open Personalization approach [39]. Portfolios may also combine user generic tasks that can be refined to specific kinds of users, e.g., [317, 437].

The persona concept has been used already in fine arts and classics (e.g., [79, 134, 571, 659]) and is applied for development of generalised user models [612]. Content chunks, entry scenes, and story portfolios have been developed in our website development team.

Context and metaphors are already discussed in Chapter 2 (see the references for this chapter).

Key Messages

Pragmatics is concerned with the contextual usage of a WIS by an individual user. **Usage analysis** comprises

- the elicitation of *life cases*, which describe user-specific stories;
- the elicitation of *user models*, which classify users by means of various profiles;
- the elicitation of *actor portfolios*, which link actors to their tasks.

WIS portfolios complement usage analysis

- by *information portfolios* capturing information need and demand;
- by *story portfolios* capturing requirements for functionality and WIS utilisation.



Categories of Web Information Systems

As announced in the previous chapter we will now elaborate on the usage aspect for the various categories of WISs that were characterised in Chapter 1. In doing so, we will emphasize the decisive features of such WISs. As we could not discuss story portfolios in detail before, these will naturally take up a prominent role in the following presentation.

6.1 E-Business and E-Commerce

Electronic business and electronic commerce (see also the discussion in [397]) are based on existing or planned business transactions that aim at the exchange of goods. Partners in e-business are businesses, administration, and customers. Actors play specific roles for these partners. We usually assume that the proactive partner is a business. The other (active) partner may be a business (B2B) or a customer (B2C). Classical subcategories of electronic trade are:

B2B: Typical examples of B2B systems deal with support supply chain management, electronic procurement, integration of finance and insurance, accounting, transport, or logistics.

B2C: A typical kind is electronic retailing. This subcategory has gained already a lot of attention and has led to a large number of WISs.

B2A: If the active partner is an administration, B2A comprises activities such as electronic procurement, licencing, and tax.

A2C: Administrations may support interaction with customers such as electronic authority, income-tax return, and forms exchange. E-government applications in this subcategory currently mainly support document exchange.

6.1.1 Branding

The pattern $\mathcal{P}^W\mathcal{U}^A$ allows a more elaborated description of electronic business using the four dimensions: \mathcal{P} roviders, goods (\mathcal{W} hat), \mathcal{U} sers, and \mathcal{A} ctions.

Within the provider dimension we distinguish between companies (B_C), personalised businesses and administration. Among companies we distinguish a broad variety of interest and corporate identity: suppliers (B_S) and classical providers such as companies or people, wholesalers (B_W) selling goods to shopkeepers or retailers, retailers (B_R) or shopkeepers selling goods to customers for their own use, and traders (B_T) selling and buying goods. Personalized businesses are based on the metaphor of business men. We distinguish between merchandisers (B_M) who are selling and buying goods, dealers (B_D) who apply a specific style of business, retailers (B_R) who sell goods to customers for their own use, brokers (B_B) who act as mediators between parties, and jobbers (B_J) who sell non-sharable services or goods. Administrations (A) can be classified into government and management.

The provider dimension is the basis for the analysis of existing business processes. During WIS development we need to compare existing business processes with life cases of users. If they match, then the development of appropriate storyboards is easy. If they do not match, we need to adapt the business processes to the needs of the users and properly reflect the users' life cases. After integration of business processes and life cases we obtain the first raw sketch of the storyboard and the utilisation portfolio.

The goods dimension reflects a large variety of goods. We may distinguish between material goods and virtual goods. Products (\mathcal{P}) are usually material goods. The description and presentation of material goods depends on their kind. Material products (\mathcal{M}) are produced, quoted, required and ordered for actors, software (\mathcal{C}) is exchanged as an electronic good, while services (\mathcal{S}) may be real or virtual. Business contracts (\mathcal{BC}) support the business, and finance products (\mathcal{F}) support the classical banking business.

Virtual goods (\mathcal{V}) are sold in order to be used by the receiving actor. We distinguish between data (\mathcal{D}) that are stream of signs, e.g., messages (\mathcal{M}), information (\mathcal{I}), i.e., data perceived, recognized, simultaneously processed, and seamlessly integrated, news (\mathcal{N}) that are displayed from various sources, and knowledge (\mathcal{K}) that is validated, high-quality and summarized data.

The development of content is mainly based on the goods dimension. The representation of goods follows a number of metaphors that are typical for the business. Depending on the kind of the good, we can use metaphors such as the shopping room, baskets for collection, leaflet presentation, or presentation of typical business cases.

The user dimension reflects a large variety of users: customers, real or virtual groups, virtual abstract people, visitors, readers, browsing or scanning readers, and zapping users. Customers require and use services or goods. They may already be buyers (C_B) or shoppers who want to buy, or customers (C_C)

interested in professional, usually long-range service, help, advice and goods. Consumers (C_K) are the typical user of an electronic business WIS. They want to use or are already using goods and thus require service, while purchasers (C_P) buy products.

The action dimension rules the development of an appropriate functionality. The word fields we discovered for the life case reflect the necessity of certain functions. The main action ^{buy} is modelled through the word field buy. This word field is extended by other word fields that are used in real-life businesses such as by ^{pay}, ^{deliver}, and ^{finance} as already briefly discussed in [712]. At the same time, these actions can be ordered. Usually, buying is integrated with payment. Delivery is scheduled for a later phase. Payment may require a user to ^{contact} a bank or the issuer of credit cards or other payment channels. The contact should be secured ^{secure-contact}. At the same time, buying and paying should preserve privacy of the customer. Delivery is often combined with a receipt of the delivery.

Additional aspects beside the brand, the business processes, and the life cases of the WIS are the intended relationship between the two partners and the system context of the WIS, i.e., the environment of the receiver, environment of the sender, the nets (local, LAN, WAN) and groupings (intranet, extranet, internet). The profile of the customer as well as intention and occasions must be taken into account, and a number of business rules may also be applicable, e.g., social, legal, and personal restrictions, and general rules of business transactions.

6.1.2 Actor Specification

The characterisation of actors is based on answers to a number of questions: Who constitutes the market segment? What do they buy and why? How, when, and where do they buy? Knowing who constitutes the market segment is not simply a matter of knowing who uses a product. Often, individuals other than the user may participate in or influence a purchase decision. Several individuals may play various roles in the decision-making process. In addition to knowing the actors, it is important to know which products target customers tend to purchase and why they do so. Customers do not purchase “things” as much as they purchase services or benefits to satisfy needs. Actors can be characterised by focusing on needs rather than on products.

Customers can be divided into consumer customers who purchase goods and services for use by themselves and by those with whom they live, and business customers who purchase goods and services for use by the organization for which they work. Although there are a number of similarities between the purchasing approaches of each type of customer, there are important differences as well.

The actors in e-business are further characterised by cultural factors, social class, personal factors, and psychological factors. Cultural factors have

the broadest influence, because they constitute a stable set of values, perceptions, preferences, and behaviours that have been learned by the consumer throughout life. Social class is also a subcultural factor: members of any given social class tend to share similar values, interests, and behaviours. They include reference groups, i.e., the formal or informal social groups against which consumers compare themselves. Personal factors include age, life-cycle stage, occupation, economic circumstances, and lifestyle. The personality and self-conception of the actor will also influence their buying behaviour. Psychological factors are based on thinking and thought patterns. Consumers are influenced by their motivation to fulfil a need. In addition, the ways in which an individual acquires and retains information will affect the buying process significantly. Consumers also make their decisions based on past experiences - both positive and negative ones.

The utilisation portfolio is enhanced by a *quality profile*. Typical quality requirements in e-business are accessibility, alert, background, community, contracts, customization, ease of use, feedback, flexibility, fun while buying, guidance, previewing, privacy, safety, security, speed, and support. Thus, the utilisation portfolio has to integrate a list of quality requirements. These requirements may be weighted for each participating actor.

6.1.3 Action Verb Fields and Scenarios

The actions used in the brand can be understood as the roots of action trees, which clarify the functionality. Actions may be given on the basis of verbs, and therefore, the main functionality required by actors within an e-business WIS are based on the word fields such as *quote*, *request*, *response*, *requisite* and *order*. These main word fields also describe the goals of the actors and can be mapped to the tasks of actors within the WIS. So, we can directly associate the activities with the portfolio and intentions corresponding to the actors. These intentions are associated with any kind of customer activities.

At the same time these word fields are associated with the word fields *document*, *be secure*, *buy*, *deliver*, *invoice*, *pay* and *negotiate*. Actors are interested in quality documentation of the products provided. They need a reliable support for security, and the provider must provide an infrastructure for delivery of goods and invoicing. If goods have to be paid by customers, then we also need a payment facility. Additionally, actors may be supported by a bargaining or negotiation facility.

The word fields result in requirements of functions that must be provided by the WIS. These functions are based on the content chunks we already developed. On the subsequent conceptual level content and functions will be supported by media types. Actors play their part within certain roles and have their rights and obligations. So, we inherit the involvement of actors from the portfolio. We can associate the word fields directly with scenarios or episodes as exemplified by the following example. Other such scenarios have

been developed by companies such as Intershop with the Enfinity Suite or Microsoft with the BizTalk sheets.

Example 6.1. In one of our e-business projects we developed the following scenarios:

Advertise and quote: This scenario allows to represent products, discusses their variations, and supports their collection. Actors are the advertiser and the customer as an observer.

Request and response: This scenario supports communication and procurement. It follows classical procurement life cases. In this scenario other actors come into play such as advisors.

Select and collect: In the selection life case the customer becomes active and the seller is a proactive partner. We may support assembling and versioning.

Bargain and contract: Bargaining and contracting is based on negotiation. Bargaining may follow bargaining strategies we observe in life cases. In a versioning and conditioning scene actors agree on a certain contract, which may include rewards.

Requisition and order: After having reached an agreement the customer may issue a formal demand, which is processed and assembled. The seller may now be in the position of a buyer if the product itself has to be assembled on the basis of other products.

Deliver and invoice: In parallel the product chosen is delivered and a bill is issued. Billing may include another e-business process such as negotiation with or handling by financial institutions. Delivery is based on an existing logistics solution on the side of the seller and on a receiving facility on the side of the buyer. Many B2C applications have failed during the 2000 hype due to weak delivery and invoicing scenarios.

Pay or return: The buyer must pay in parallel and the payment must be acceptable for the seller. This may involve another actor as arbitrator. Payment usually concludes the buying act, though the return of goods, raising claims, and requiring maintenance or extensions of goods have to be taken into account leading to extensions by scenarios that support processing of the goods, inventory of goods, tracking of use of the goods, etc.

Each of these scenarios can be complex. They can be formalised using the techniques developed in Chapter 3.

6.1.4 Elicitation Strategy

Story portfolio development may be simple as long as the e-business WIS is simple and will not evolve in future. Since most e-business systems will evolve and this evolution must be handled anyway we may propose the following strategy for utilisation portfolio elicitation, which was applied in one of our e-business projects:

1. First elaborate on the opportunities for the development of the e-business system. This *portfolio scoping* is similar to scoping in Software Engineering.
2. After knowing the main business life cases identify restrictions of the envisioned system. These restrictions are mapped to *production rules* which guide the development and implementation of the WIS.
3. Finally derive the utilisation portfolio in detail.

Scoping

Portfolio scoping is based on the following activities:

Analysis of customers' business along value chain: Not all customers have the same value for the business, so we may be interested in some customers and not so much in others. This analysis can be based on an opinion pool of the subdepartments of the seller.

Determination of customers' business vision and strategy: Customers have their own targets, objects, objectives and aims, based on which they are likely to accept the products and business strategy of the seller.

Determination of customers' business model opportunities: Customers may have more than one opportunity to obtain products like the ones the seller offers. So the competition on the market is taken into account while developing the e-business solution.

Elaboration of potential business scenarios and interaction space: A number of business scenarios can be chosen from real life cases or envisioned life cases. As we cannot support all these, we choose the most appropriate ones.

Analysis of marketing aspects: Marketing aims at supporting the profit target by promoting sales and distribution of a product or service.

Development of an overview on potential solutions: As there may be a wide range of different solutions, the utilisation portfolio should be restricted to the ones with the best chances of adoption.

Customer strategy definition: The utilisation portfolio must be acceptable to the customer. As accepted life cases have already been developed, we can compare them with the envisioned ones and choose the fittest.

Deliverables of portfolio scoping are extensions of the life cases that are incorporated into the utilisation portfolio. These concern customer intention and strategy, i.e., the involvement of customers is modelled based on their intentions and strategies, customer business focus, i.e., the tasks and goals of customers are used for deriving the focus of customers, business model opportunities, i.e., the business model is based on the utilisation portfolio we derive, variants of scenarios within the interaction space, i.e., additionally learnt scenarios are used for modelling side scenarios or for further development, and questionnaires for the next steps, i.e., the decisions made are recorded as

guidance for the open issues in further development. The final result is the specification of the utilisation portfolio expressed by business scenarios and integrated into the interaction space.

Derivation of Production Rules

In a second step we can analyse the e-business life cases and derive *production rules* for later use in the WIS development on the basis of the utilisation portfolio. These production rules require a deeper analysis of the WIS portfolio:

Analysis of business life cases: Business life cases have their interdependencies, their specific support functionality and their specific content. We analyse these life cases and determine which properties must be supported by the WIS and which properties may be neglected.

Determination of core business life cases and their requested functionality:

Due to the restricted resources for development we determine priorities among life cases.

Service architecture definition: The interdependencies among business life cases restrict the ability to completely separate aspects. Thus, consideration has to be given to adhesion among scenes, functions and data support. A service architecture proposal that preserves these adhesions can already be sketched at this stage.

Data volume definition: Data exchange through communication channels may be a bottleneck and a security risk. For this reason we need to know the communication data volume per customer, participant, product, and per scene, respectively.

Derivation of potential environment and components: Once we know the life cases in depth and the requirements for data exchange and customer support, we can derive proposals for a potential system, which may consist of several interacting or collaborating components.

Derivation of an implementation proposal: Finally, these findings may be combined within a number of production rules as a proposal for the implementation of the e-business WIS. This proposal is not the final one, but permits better coping with scenario development and data type and function specification.

The analysis of the life cases and the derivation of the utilisation portfolio allow us to derive a number of proposals that ease development:

Customer business scenarios with core business life cases: The development may first be concentrated on the main or core business life cases. For these life cases we develop scenarios based on the techniques discussed in [727].

User interface layout and navigation: After analysis of the core business life cases and development of the corresponding utilisation scenario we can sketch the general layout of the user interfaces and the general structure of

navigation. The navigation structure must capture completely the scenes of the scenarios we specified so far. The user interface layout may result in a development of so-called mock-ups.

Functionality Requirements: As the utilisation portfolio is now known for the core business life cases, we may now collect the functions that are necessary for it. Typically, these functions may be generalised to function families representing similar functionality.

Interfaces: The life cases permit the identification of a number of interface software tools and hardware devices. As e-business systems are to support all customers, we need all these kinds of interface tools.

Service architecture: The utilisation portfolio directly provides an understanding which application part is called at which moment. Therefore, we may now derive an application architecture for the service, which provides a rough impression on the envisioned architecture of the WIS.

Environment and components: After capturing the life cases and during elicitation of the utilisation portfolio we collect also information on the potential usage environment and components that must be compatible with the system. This leads to a proposal for the technical infrastructure.

Questionnaire for next steps: As the development and the requirements elicitation process will never be complete, we may use a questionnaire for collecting all open questions that must be answered in the sequel.

The result of this analysis is a set of production rules for further WIS development, which will ease decision making during the next development steps.

Defining the Details

The utilisation portfolio is finalized by a *utilisation portfolio definition*. Main components of the utilisation portfolio we gather are the following:

Detailed business process model: The business processes are specified on the basis of scenarios and their interdependencies.

Integrated services definition: We define all details for functionality, interfacing and content support.

Environment and WIS context specification: We define the differences between requirements and components on hand and the components that still must be developed.

Business life cases that can be supported at an earlier development stage: As a project plan for delivered components is developed, we can also derive a stepwise procedure for WIS installation.

Requirements definition: The utilisation portfolio may serve as a light-weight requirements definition.

Technical and architectural feasibility: The utilisation portfolio is accompanied by a proposal for the technical realisation of the WIS.

The final result of this gathering process are the utilisation portfolio itself, production rules, scenarios, a content acquisition and publishing strategy, a detailed systems integration concept, the environment for the WIS, a proposal for components and their customization, and finally an estimation of the development effort including implementation time and project costs. The result of the development is the utilisation portfolio in all necessary details.

The organised process of story portfolio gathering has a number of advantages. We use a transparent modelling method that is derived from project experience and from real or envisioned life cases. The utilisation portfolio obtained in this way fits seamlessly into the business strategy and the marketing aspects. The business scenarios include business life cases and are at the same time linked to ideas for the implementation. The loss of information is reduced. As the approach is centred around the life cases and the business processes, it provides a common base for communication and documentation. The development model supports discussion and decision foundation by incorporating different model views. Project and development risks will be minimised. Development speed can be increased and consequently the time to market is decreased. The resulting system developed is easier to maintain and faster to adapt. During development reference models of business scenarios can be developed in parallel leading to shorter implementation time for follow-on projects.

6.1.5 Supporting Features

These WIS portfolios can be used for deriving requirements for content and functionality. In the case of an e-business WIS we are interested in the presentation of a product, ordering, sale, distribution, payment, etc. Content presentation depends on the kind of e-business. The utilisation portfolio may result in additional development of functionality. We can distinguish four kinds of e-business utilisation portfolios:

Supermarket-type services: E-business WIS supermarkets adopt utilisation portfolios that are typical for department stores or mail orders. The minimal content required is product presentation, information on the business activities, tips and tricks for the products, news, and hits. Therefore, additional scenarios are required for maintaining and supporting customer rooms, chats, (advisory) services, and discussion groups. We may distinguish between chains for direct customers, configuration sale, and direct sale. The first service subkind supports fast/speedy casual customers as represented by visitors of malls. In this case, we need also a support for advertisement. The second subkind must be supported by full information on all possible variants and configuration functionality. The service requires facilities for building packages, and in some cases these packages are assembled from products from different suppliers. Direct sale (or reselling) scenarios follow an auction style, and often adopt a first-come-first-served policy.

Sales agency services: The utilisation portfolio requires a restricted product comparison specialized on categories. It includes scenarios that are intermediary for other sales agencies or support reselling. Therefore, additional content provides information on products, supplier relations, and banking.

Sales broker services: The utilisation portfolio is enhanced by product comparison, e.g., for product categories, suppliers, and special offers, and oriented at the customer. It needs specific brokerage and search functions. Additional content is required for the management of information brokers, logistics, product/content management, and customer relations.

B2B WIS: The utilization portfolios are mainly a mapping of typical business life cases dedicated to closed communities. Therefore, these WISs require a management of sales activities, contracts, and customers.

Business Culture

The storyboard of a WIS may be restricted according to specific business approaches such as business culture. The Japanese style of business activities differs significantly from the European or Arabian styles. This requires the explicit specification of *activity patterns* (or generic workflows) that are applied in the business. For instance, the activities provided for C2C WIS, for B2A WIS, and for B2B WIS differ to a large extent. Therefore, we can limit the utilisation portfolio to those activities that are common in the area considered.

Patterns typically observed in business activities are based on the form of the activities, the roles the partners play in them, the initialization process, the summarisation or termination process, the space and time limitations, and the general rules applicable to the business. In addition, details on the style of business may be supplied such as the typical discourse and the workspace to be provided.

Typically, activities are based on the collaboration of actors, which is determined by a *collaboration style* specifying the environment such as supporting programs, the style of cooperation, the kind of communication, and the coordination facilities, and a *collaboration pattern* specifying the roles of partners, their responsibilities and rights, and the protocols they rely on. The collaboration style reflects the context of the activities, and the collaboration pattern reflects the way how the business is acting. The collaboration style and pattern are mapped to functionality that is requested for the WIS.

Example 6.2. SAP R/3 uses a number of business activity patterns. [Figure 6.1](#) displays a direct sequenced pattern. This pattern is governed by the life case “trade-with-payment-in-advance”. Another applicable life case is “trade-with-payment-after-delivery”.

Business patterns follow a number of styles. A typical style of business for C2C WISs are “trade-but-do-not-trust”, i.e., customers have to pay first

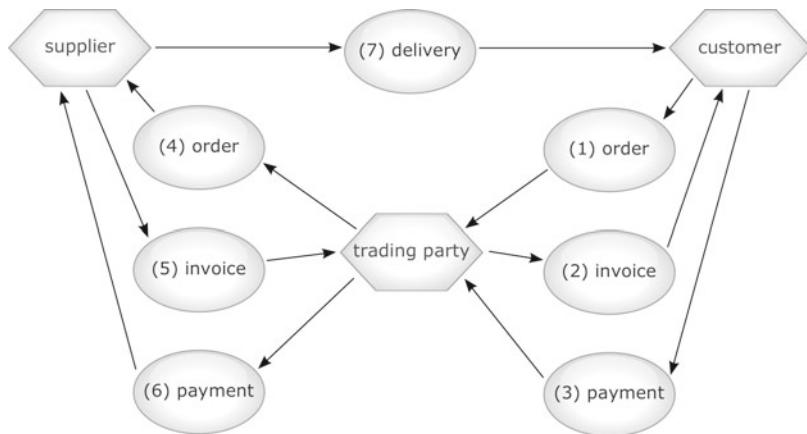


Fig. 6.1. Business activity pattern: sequenced business

before they obtain the product. B2C WISs also use the style “trade-and-trust-but-be-in-control”. These styles are enhanced by actions. These activity styles closely correspond to collaboration styles and collaboration patterns with the customer in the role of a trusted, not-yet trusted or untrusted partner.

Business patterns are governed by the organisation structure of all parties involved. The organisation structure of suppliers and traders is often hierarchical. Business may involve more parties such as trusted third parties, producers, and deliverers. The specification of parties is based on the specification of actor profiles and portfolios. Business styles are influenced by logistics decisions, purchasing organisation, sales approaches, supporting infrastructure, partner enactment (banks, credit card organisations), and general restrictions for the delivery of goods. These already influence the activity style and rule the business pattern. The business style and the business pattern are enhanced by styles and pattern of procurement, stock, and distribution. These processes are related to consumer response pattern such as replenishment, assortment, promotion, product introduction, retailment, and category management.

WIS Portfolio Template

The discussion of story portfolios for e-business allows us to combine properties into the following WIS portfolio template:

WIS portfolio:	<portfolio name>
Categories:	<list of categories,> <possibly with subcategories>
Refined brands:	<general description>
Kinds of WIS portfolio:	<general description>
Partners:	<list actors with portfolio>

	⟨possibly with rights, obligations, roles⟩
Scenarios:	⟨description of the scenarios used⟩
Activities:	⟨list of word fields characterising activities⟩
Activity style:	⟨general description⟩
Activity pattern:	⟨general description⟩
Collaboration:	⟨style and pattern description⟩
Word fields:	⟨content, functions, and context⟩
Supporting content:	⟨content chunks⟩
Content portfolio:	⟨demand, consumption, production⟩
Supporting functions:	⟨function chunks⟩
Functionality portfolio:	⟨demand, consumption, production⟩
Non-functional requirements:	⟨quality requirements⟩
Context:	⟨derived specifics of the context space⟩
Based On:	⟨life cases, portfolios, profiles, intentions⟩ ⟨scenarios, general tasks, audience⟩ ⟨mission, goals, brand⟩ ⟨business rules and processes⟩

We notice further that electronic business and electronic commerce are evolving towards more sophisticated utilisation portfolios. Currently, they are based on existing contracts and products, and aim at decreasing costs, increasing delivery and reaction time, achieving better pricing, increasing competitiveness, and obtaining higher product quality. However, new business models will be developed that are based on global presence and selection. The content on display will move to customized products and personalised services. Therefore, utilisation portfolios will be based on new businesses, products, and services, which can be agent-based and agent-supported. They will use supply nets instead of chains, intermediaries, and perishable virtual companies. We already observe a shift from classical products towards new products (e.g., in logistics www.tnt.de) and new markets (the classical example is www.amazon.com). A unification of both directions is likely based on information brokering (e.g., www.paperball.com), recommendation services, or new market places (e.g., www.jango.com). For this new business, rights, and property models are needed, which require adjustable and self-adapting utilisation portfolios.

6.2 Communities and Groups

Communities and leisure groups are gathering places for registered members, who form an interacting community of various actors with common location, intentions, and time. They are based on shared experience, interests or conviction, and voluntary interaction among members towards common goals. In the case of work communities they deal with issues affecting the legal profession,

and are concerned with furthering the best interests of their members. Communities often employ leadership and hosting. Membership requirements and obligations, resources and amenities, member characteristics, community policies, and activity style and pattern create a large variety of community/group WISs.

Communities are member-created and often based on leadership and on hosting. They can be work-related, in which case they deal with issues affecting the legal profession and are concerned with promoting the best interests of its members. Such communities often serve as lobbying agencies. They may also support a hobby or any other pursuit outside members' regular occupation. Communities provide a flexible code of conduct and a wide range of roles.

Community members often offer their contributions, advice, reviews, and emotional support without expecting direct reciprocity or financial reward. Intentions behind this altruistic behaviour can be quite complex, as they include commitments to the cause or interest associated with the community, the desire to help others, expected benefits from displaying experience and knowledge, and personal satisfaction or self-esteem derived from acting within a community.

6.2.1 Branding

The utilisation portfolio is based on the community or group member profiles. The general brand $\mathcal{P}^W 2 \mathcal{U}^A$ must also be specialized for community and group services. The provider and the users may coincide for community WIS.

The actions depend on the kind of the community or group. The life cases of these communities and groups vary very much. Nevertheless, we can identify several types of typical scenarios. Active scenarios (^{act}) are activities by the receiving actor(s). Information scenarios (^{inform}) or more specifically ^{ask} and ^{search} are mainly governed by the provider. Participation scenarios (^{attend}) request the interaction of receiving actors. Community and group WIS are mainly following the Group2Group^{act} or the Group2Visitor^{inform,act} brand.

Discussions are another kind of action. We may distinguish *chat-oriented*, *collaboration-based*, and *topic-centered* discussion groups. The first group does not limit contributions. Main actions are initiate, respond, inform, summarise, and close. Collaboration-based discussions follow a collaboration pattern and preserve a collaboration style agreed in advance. The main metaphor used is the “meeting”. Main actions are open, close, propose agenda, issue discussion, contribute, and inform. Topic-centered discussion groups are based on a topic and preserve dialogue acts. Actors may be authors, responders, organisers, or supporters. Authors are typically classified and have a number of privileges, while responders respond to some data and use a response pattern and preserve a response style. Tokens are a convenient vehicle for support of topic-centered discussions often generalising communication protocols.

Example 6.3. The process for finding a group decision is typical example for topic-centered discussions. For instance, members of a program committee of a conference collaborate during decision making, whether a paper is to be accepted or rejected. They act either as reviewers or as ordinary members of the program committee. Reviewers of one paper may revise their review, comment on the reviews of other members, and exchange messages within the subgroup of the program committee comprising the reviewers. An ordinary member of the program committee may comment on the reviews and request a change of evaluation.

All members will make a common decision that depends on the other discussion topics, i.e., on the decisions for all papers. Opinions may be stated with a certain confidence. Opinions may be distributed to all or selected members of the program committee. Authors of papers are excluded from all discussions on their papers. The actions of a member of the PC may be mandatory, optional or not permitted. So, we specify these obligations and rights using deontic constraints.

6.2.2 Actor Specification

Actors of communities are members with a variety of involvement. Some of the members may act as fellows or heads of groups or of subgroups. Communities may be partially open to all visitors and often operate as integrated institutions. They promote, organize, and regulate competitions, coaching, and training and may holds regular meeting and also competitions.. They may have a disciplinary power over their members. The behavioural pattern of members in communities and groups, their collaboration style and their collaboration pattern may vary.

Actors usually derive more than information benefits from the community. They may derive social, health, and professional benefits that come from interacting with other people who know things better than they. Active can be active, proactive, or passive. The benefits are usually directly related to the level of activity of members.

Example 6.4. The W3C community is perhaps the largest active community WIS. It uses sophisticated document management, a large variety of groups, and is based on contributions of authors.

Authors may need a document-specific policy. Contributors to discussions and forums may want to provide contact data to their fellow collaborators but not to the public. At the moment, the best way to accomplish this is to not provide any contact information within the document and hope that people already know how to reach the contributor. Internal organizational groups may need their own group-wide policy. The W3C community site oversees a number of working groups that deal with specific interests concerning the Web. These groups are composed of W3C employees and member organizations' employees. Their work is available to all member organizations, some to the public

as a whole. However, some instances may arise where only group members would be allowed to see certain information, and the W3C database currently only addresses groupings in terms of membership organization employees or W3C employees. It can but does not use a working group perspective of its entries due to scaling issues.

Some grounds for negotiation must be provided with these separate parties trying to describe their own policies on the same data. It is possible to generate an acceptable document policy to all parties involved since it is not necessarily feasible for all of them to meet and negotiate in person. There is also a need for negotiation between people requesting data, the clients, and publishers of data. Sometimes a corporate policy must be kept private, for whatever reasons. The concealment of policies such that they cannot be reverse engineered easily is an important component.

Lastly, the content schema needs to be extensible to any future changes in the W3C community.

Web2.0 sites provide other examples of enhanced community/group WISs that are even more user-driven. Examples are [GoogleActSense](#), [Wikipedia](#), [Flickr](#), etc. Communities may share their opinion or beliefs or partial knowledge (e.g., [Wikipedia](#)), their diary or visits (e.g., [del.icio.us](#)), their intentions of internet utilisation (e.g., [43thing.com](#)), and more. While these communities are currently more or less leisure communities, they will evolve into large bodies requiring lead enhanced utilisation portfolios for communities.

In particular, support for collaboration and participation will be needed. Typical forms of participation are already practised in forums. As collaboration requires coordination, communities start to establish contracts, which members must keep to, and penalties in case of contract violations. For this a regulation institution is required that forces collaborators of communities with contracts to follow the contracted rules.

This gives rise to *collaboration utilisation portfolios*. Collaboration means to work together towards a common goal, i.e., it combines of three aspects:

Cooperation takes parts of actions into account that jointly define a group action.

Coordination of actions of individual actors is necessary to achieve the desired result in a cooperative action.

Communication between actors involved in the same group action is indispensable for coordination. It appears in a variety of facets, e.g., as an act or instance of transmitting a verbal or written message, a process by which information is exchanged between individuals through a common system of symbols, or as a technique for expressing ideas effectively (as in speech) using the technology of the transmission of information (as by print or telecommunication).

6.2.3 Verb Fields, Functionality and Scenarios

So, the main word fields associated for the utilisation portfolio are the following ones:

Contribute: A member of a community gives or supplies a document to the community and shares it with the community subject to certain obligations and responsibilities. Obligations include modification obligations such as updating and removing the document. The member of the community becomes an author and has a role as owner. The community, however, obtains or possesses the contribution. Responsibilities of the author include quality obligations for the contribution.

Join: Users may join a community and thus obtain a role of a member of the community. The membership requires the commitment to the coordination style and coordination pattern of the community. This commitment implies a certain behaviour and collaboration with the members of the community. For instance, members agree on regular occurrence or logical relations. The membership may be public or hidden to visitors.

Meet: Members of a community come together at a particular time or place, entering into conference, argument, or personal dealings with other members of the community or the general audience. During the meeting they receive or greet in an official capacity of the community. Members may form coalitions before, during and after meetings.

Organize: Becoming a member of a community brings rights and obligations. The member may be permitted to develop a unit within the community and has a function during this development. In this case the member becomes a coherent functioning whole, and contributes with his or her own profile and portfolio. Organisations often require arranging by systematic planning and united effort.

Publish: The author makes his or her contributions generally known. The community decides whether the contribution can be disseminated to the public based on the rules of the community. The contribution is then produced or released for distribution.

Subscribe: Communities often have their own publication lines. A member subscribes to some of them according to the rules of the community. The subscribed services are (periodically and regularly) received by the actor. The member of the community agrees to purchase and pay for securities. The subscription can also be extended to new offerings.

Withdraw: The members of a community should have the right to cancel the membership of the community or part of the subscriptions. In this case the user may lose the ownership of contributions he or she made. The withdrawal may require particular official procedures.

Additional word fields supporting the activities of communities are *work in the community, collect, founding, help, promote standards and beliefs, and protect the interests of its members*. Groups and communities are interested

in keeping their members informed, in providing a guidance, representing their profession, and in advocating their own views. These word fields have a more complex application than the word fields used in electronic business. We cannot apply them sequentially but must provide facilities for their parallel application. They can be repeatedly called, so the scenarios that can be based on these word fields include recursion.

Classification of Community Portfolios

We may distinguish between two specific kinds of communities that lead to different scenarios and different behaviour of actors:

- *Transaction communities* are oriented towards markets (e.g, buying, selling), work places or based on a call center paradigm. The main content used is based on product representation.
- *Fora* typically use a simple order mechanism (chronological, time, topic) and support discussion groups for (a)synchronous discussions. Typically they provide additional links that support their work. The content is concentrated around the discussion state and the results achieved so far. Additionally, the community might be interested in archived results. Therefore the main activities supported by the WIS are information management, scheduling of forums, archiving, splitting into groups and joining groups, and providing visibility. Additionally, partial anonymity is required.

Online communities offer a number of benefits to their members, to third parties, and to society. At the same time they may harm their members, third parties, and the society. Since members spend their time within the community they may lose relationships with the real world. Intellectual property infringement, trademark infringement, and erroneous content are unsolved issues for proper development of community WIS. The security risks, threads, and vulnerabilities are as difficult as for business WIS. Controls should be provided for the security of sensitive information wherever a user has contact with the data. They must also detect and report possible security violations. The systems development must also consider privacy concern. Membership lists are often attacked. Reports on hacking logs of systems appear monthly. These risks and threats are constantly evolving due to the development of the technology.

Transaction Communities

Transaction communities such as vger.kernel.org, www.wikipedia.com, www.dama.org, and www.pgdp.net use the WIS to (voluntarily) produce real products, for instance, software, literary works, or other creations. Typical transaction communities are the open software groups. Members are likely to be deeply involved. These communities are at the same time not as enduring as other communities.

The SeSAM system we report in the special community part of the book is supporting parliamentarians for their specific collaboration, their communication with inhabitants, their sessions, etc. A parliamentarian may play a number of different roles, has a variety of parts, is involved in a number of tasks, may form coalitions or interest groups, and has often to consider time limits and social constraints.

The scenario of transaction-oriented community WIS may be governed by *agendas*. In this case, a temporal dimension is added to the way how communities are acting. Communities often do not have authorities who certify the accuracy of the contribution. Instead members contributing have the responsibility for quality, accuracy, completeness. Other communities may have their reviewing or voting procedures.

Early transaction communities have been geared towards support of specific group tasks such as shared calendaring, emailing, scheduling or towards integrated support functions such as dispersed project management. These systems have now evolved to shared databases. Currently, community WISs use the most advanced functionality compared with other categories of WISs.

Forum Communities

Despite the differences across types of shared interest, fora are characterised by anytime-anyplace communication with weak social context cues, aggregated small or micro-contributions, and norms of interaction. The boundaries across types of forum communities can be fuzzy. We may, however, distinguish five types of forum communities and their specific scenario:

- Consumer communities (e.g., www.audifans.com or www.lugnet.com) are based on common interest in and are loyal to a particular brand, team, entertainer, politician, or media property. Users voluntarily share their information and passion with thousands or hundreds of thousands of others. They share their experience, e.g., their information about parts suppliers, tricks, and mechanics. Comments may be positive or negative. Consumer communities are typically characterized by certain instability. Although the interest may be an intense one, the intention to join the community and the activities of the members typically represent a fairly small and often small portion of a member's life.
- Avocation communities (e.g., www.bikeformus.net, www.chessclub.com, and www.LambdaMoo.info) attract experts and enthusiasts who form and join voluntarily the forum with the objective to increase their pleasure and proficiency in their hobbies, leisure, or work. The product may be a means but is not of primary interest of forum members. The main content is based on 'how-to' data. People share their interest and their expertise. They may organise tournaments or competitions. These communities are relatively stable since they represent a relatively enduring part of users life.

- Place-based communities such as www.bev.net and www.MeetUp.com are organized by and for users who live in a particular locale. They allow residents to improve social connections, to participate in political and other processes. They are relatively stable since the shared interest and object last at least as long as members reside in the physical community.
- Common condition communities (e.g., www.seniornet.org, www.systers.org, and www.deja.com/group/) share their experience and interest of a common condition. The purpose of members of these communities is to learn how others experience or are coping with their conditions and to share their own experiences. Members may be alumni of a particular organisation, may have common demographic, medical or psychological conditions, or want to overcome loneliness. The common condition community is usually very stable and long-lasting.
- Concern communities such as www.moveon.org, www.419legal.org, and groups in www.deja.com/group/ share the members interest in common political, social, or ideological concern. These communities are often hyper-active and claim their omnipotence and omniscience. Members have multifaceted ties to that world. They may organise real-world events, letter-writing campaigns, rallies, fundraisers, etc. The shared interest of concern communities is likely to be a deep and abiding one for their members.

Functionality and Derived Scenarios

The functionality required for micro-contributions includes the management of threads and the aggregation of these contributions. Aggregation allows to combine micro-contributions into larger contributions. It is necessary for efficiency of interaction and for social effectiveness. The simplest form of aggregation is the attachment of a contribution to others, e.g., responding on a number of contributions. A thread is a seed message combined with all reactions on it. The simplest forms are questions with replies and proposals or statements with comments. Community WISs may have hundreds of threads active at the same time. Threads can be ordered by annotations or topic maps. These maps use their own structuring, e.g., hierarchical structure, category-ruled structure, or more complex poly-hierarchies.

Entire documents may use a very complex structure, may have internally displayable different versions and configurations, may have their history and evolution, and may be associated with users by a variety of ownership relations.

The functionality of community WIS includes also facilities for rating, ordering, discharging, or voting. Rating can be based on reviewing of contributions or parts of them. Additionally, community services use functionality features such as:

- Sophisticated graphical user interfaces support all kinds of users ranging from handicapped to internet-age educated users. Systems take advantage of color graphics and sound capabilities of modern computers.

- WYSIWYG editors support document evolution for all users independently of the system context they use. They offer more comfort, security, and privacy than text processing systems currently employed.
- Community services are not used on their own. Therefore, integration of components of different applications into a standard interface provides a WIS like a multifaceted tool.
- Online communication and collaboration support active discussions and interaction among participants. They allow textual and video exchange in real time.
- Replication and synchronisation support members with an access to the WIS and replicate the data locally in the system of the member and with synchronisation of editing efforts of members. Update conflict management is based on a clarification scenarios with all involved parties.
- Remote access and update is supported for all users based on the replication and synchronisation subsystem.
- Scenario management supports the completion of tasks of one member based on the agreed-upon coordination rules independently on the completion of other tasks by other members.
- Community WIS offer group calendar diaries and project schedules to help members to coordinate their time depending on the role.
- Secretarial functions are background filtering and ordering functions that help members with ‘clean-up’ and ‘overhead’ functions. These functions are based on information filtering mechanisms.
- Project management functions support controlling management of documents, schedules, and personnel. They provide collaboration among members, among groups, and among communities without regard of their location or time through a variety of forms such as forums and emailing. Therefore, the right information gets to the right people at the right time.
- Blackboard facilities support tracing of tasks, their completion, members’ involvement, and scheduling.
- Report functions are integrated into the blackboard so that any member and also non-members can survey the state of affairs. Shared whiteboards are either stand-alone copy boards, peripheral boards with data transfer to all collaborating clients, or interactive boards with internal functions for linking, contributing and accessing for all users.
- Collaborative portals are a single point of useful, comprehensive, ubiquitous, and integrated access to customised and tailored content. Members can conduct their own communities, interests, tasks, portfolio, and job responsibilities. They can manage personal data, create personal workplaces and workspaces, have real-time chat capabilities, and increase collaboration capabilities based on tracking facilities that provide information on active members and their activities.
- Asynchronous collaboration is supported by newsgroups, list servers, workflow systems, group calendars, and delivery systems. The latter keep track

which data has really reached the member and which data transfer is still pending.

- Synchronous collaboration is supported by chat systems, videoconference facilities, specific member relationship functions, threaded-discussion systems, virtual workplace functions, collaborative writing functions, and mobile messaging functions.

The functionality and scenario will extensively change in future. We already observe the transition of the information society or information age to the *participation age* or participation society. The technology on hand changes social habits of community members. The information becomes distributed in the network. Privacy and security are thus becoming the crucial quality properties. At the same time, community systems will be standardised. In this case they accommodate collaboration on the fly, on demand, or on leisure.

6.2.4 Content Chunks

Community services need to be enhanced by sophisticated content chunks such as workspace and workplace. Often communities create, develop, finalise, and exhibit their own documents.

Content chunks important for community WIS range from micro-contributions to entire documents. Micro-contributions are individual contributions of relatively low size, complexity, and lifespan. Their production and posting does not require large efforts. Community WIS may however have barriers for distribution of contributions. The content semantics is rather low since it is difficult to achieve nuanced understanding through micro-contributions.

The content is often profoundly personalised. Its form and structure are personal, e.g., personal thoughts and experiences. Questions, requests, and replies are based on an ortho-normalised language. They are framed for human understanding. They also reflect the situation of members of these communities. Personalised information can increase pleasure or competence for some members and can challenge the beliefs and assumptions of others.

The content personalisation leads also to erroneous, malicious, or destructive information. The peer review of content may dampen this kind of harm but cannot prevent it entirely. The classical publishers business has shown that communities must form their own authority that assures quality. Most developments in the open source software area or open source dictionaries area do not have their authorities. For this reason content must be also critically received and interpreted by the reader.

The quality requirements are different from those in e-business: access control, distribution, feedback, flexibility, (partial) mentoring, portability, (partial) privacy, reliability, safety, searchability, and security. They are extended by typical community quality requirements: to provide an achievements orientation, affiliations, self-actualising environment, humanistic-encouraging content, good interaction style, rational decision processes, clear criteria, free flow

of information, inclusion, openness, non-competition, non-opposition, synergy, constructive behaviour, and knowledge. At the same time, community WIS must overcome problems that can be observed in badly organised communities: poorly defined goals, poor planning, hidden agendas, conflicts, ignored alternatives, groupthink, lack of knowledge, poor culture, distraction, missing information, poorly defined criteria, passive behaviours, aggressive behaviours, mobbing, and fear of speaking.

We discuss in detail community services in the last part of this book. For this reason we can restrict the development of utilisation portfolio to these general remarks. We consider community websites as the main kind of websites of the future. So far, community websites only represent the intention of well-defined and stable communities such as associations. We may now observe another kind of community site: entertaining communities, learning communities, blog communities, and marketing communities. Typical developments in this direction are blogs, podcasting, tagging, and wikis. Websites are still often author driven and follow the publish/provide_story/support and advertise/wait/attract/react/retain patterns. Users follow mainly the inform/subscribe/obtain/answer/come_back pattern.

A typical metaphor used for community WIS is the spatial metaphor. Topics are associated with rooms. These rooms have their own agenda and contributions. For instance, contributions may be organised around civic functions such as gardening, libraries, and laboratories.

6.3 Entertainment and Gaming Systems

Web-based entertainment systems are less data-intensive than systems in the other categories of WISs. They are therefore less important for us, but we include this brief discussion of entertainment WISs for the sake of completeness.

Entertainment systems aim at evoking feelings and thoughts. They are usually combined with (personal) e-business sites. Sometimes, they are also embedded into educational sites. They are similar to real life entertainment. The general brand P^W2U^A is specialized by active scenarios (^{act}) with activities taken by the receiving actor(s) and by the products of entertainment. The products are usually associated with fun (\mathcal{F}). Therefore, the brand of entertainment (fan) sites is given by $B^{\mathcal{F}}2V^{act}$ or $V^{\mathcal{F}}2V^{act}$ with B for business and V for visitor.

Entertainment sites suffer from incompatibilities and unstable technology. As CD-ROM sophistication has dried out all singleton player entertainment sites, the emphasis is now on interactive or collaboration games. The aspect of collaboration is analogous to community/group WISs. The placement of functionality and content follows however the gaming approach, i.e., links to a guided tour, room or apartment metaphors for placement of items, exhibits, and setting expectations very well before breaking the rules.

Entertainment sites can be classified into a number of kinds, genres, and classes such as head and expert, develop, or fight. Kinds such as tournaments, n -player games, board games are specified by a number of criteria that are not subject to interpretation or understanding of the user and are not dynamic. Genres classify the experience the player gains, e.g., reasoning, reacting. Classes characterize the storyboard of the game, e.g., development, deduction, shooting, trading. Additionally, the storyboard has in entertainment a special situation: conflicts and conflict resolution.

Strategy games are the most important entertainment WIS beside action games, adventure and role games, sport games, and puzzle games. Since strategy, tactics, and role games may also be used for edutainment we discuss these in more details. They are often based on puzzle-solving scenarios where a player has certain targets such as a renaturation or area development target. The space of applicable steps has a high dimensionality. Applicable steps are dependent on each other and have a number of re-, rely-, guarantee- and post-conditions and distant and side effects.

Scenarios can be based on word fields such as *decide*, *explore* and *detect*, and *ask*. They represent often complex, network and dynamic situations. The complete information on the situation cannot be retrieved by the player. Player must often follow patterns of causal and hypothetical reasoning, i.e., development of targets, collection of data and development of a model, analysis of the current situation and its opportunities, prognostic elaboration and extrapolation, planing of activities, and control of the effect. Targets and goals may have positive or negative effects. Sometimes, negative effects are preferable. Goals may be preferable or avoidable, general or specific, simple or complex, and implicit or explicit. Data have their own characterisation ranging from important in the given situation to negligible.

This gives rise to a number of utilisation scenarios. Games are specified by

- the intention of the game by a variable and quantifiable outcome that can be valorized, e.g., the purpose for the player, the intent of the player,
- a number of transition rules with content parameters, with function parameters, and with states to be transformed,
- the player effort for the game depending on the profile and the portfolio of the player, e.g., abilities, education, tasks, collaboration, and
- the context of the game, e.g., hardware and software context, actor context, storyboard context, temporal context, organizational and social context, and provider context.

The attachment of the player to the outcome may change the game behaviour. Consequences of steps the player is taking may also be under negotiation. Games are mainly flat and linear scenarios that follow a game story. Additional information, tricks, tips, customer treatment, contracting, workspace supply, communication support, group and game rooms, help/tips, downloads, payment, conditions of usage extend the playout of the scenarios. Hobby sites use scenarios similar to information or community sites. In this case, hobby

description, links, documents and other media objects are provided. Fun and humor sites often do not have any scenario. They mainly consist of links and documents.

The storyboard of games and scenes can be generally specified by the scene intention in [Figure 6.2](#). Players of games do not have the full control and the full knowledge of the state of the art. At the same time, they have the illusion of full control, of full vision, and of full functionality.

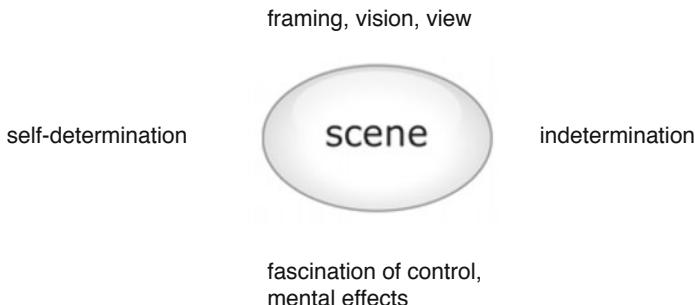


Fig. 6.2. The characterisation of the intention of scenes within entertainment

Functionality of entertainment WIS is often very sophisticated. It ranges from simple click and point functions to story instantiation or story restructuring functions. The navigation is based on simple tree or graph structure whereas the structuring depends on categories/rubrics or ontology. Personal entertainment sites, advice sites, and survey sites also do not have any scenario. They follow mainly approaches we discuss for corporate identity and forum. Their navigation structure is rather simple, often only a one-level structure. They follow metaphors of show-business presentation. Similarly experiment presentation sites are built. Special entertainment sites provide a group service, use a pay-per-view payment and are based on journal or presentation metaphors. They provide a collection of media data, customer data, and contracts. Only the last case can be described through word fields such as *survey*, *billing*, *condition*, *order*, *member service*, and *linking*.

Entertainment WISs are based on a sophisticated actor specification. Actors are often highly demanding in their requirements. Their quality requirements include accessibility, art/mechanics, downloads, easy exploration, feedback, (sophisticated) functionality, guidance, immersion, integration, and (narrative) storytelling.

Actors are associated to a portfolio that consists of tasks of certain degree of difficulty, with certain effects, duration for completion and benefits and penalties for their completion. Tasks are structured, ordered, and prioritized.

Context is often deeply elaborated in entertainment WIS. They are providing all illusions and functions known for virtual reality.

Since virtual reality is considered to be an image of the reality, it is also characterized by history, folks, organisations, realms, and ambitions for further development.

The results of execution of a scene are related to tasks currently under consideration with their pre-determined plans and scenarios, with time management, with management of state changes, with resources, functions, and collaborating actors. The tasks are structured and ordered and follow a certain plan. For their solution a number of resources must be provided and a level of resistance must be overruled. Activities can be described by word fields for all major verbs in our language. Tasks may be open, completed, under completion, under repetition, etc. Entertainment WISs usually reward players for successful execution. Scene change follows general stories envisioned.

The involvement of actors in task completion is pre-specified on the basis of *characters*, i.e., the roles an actor may potentially play, the part the actor plays within the scenario, rights an actor obtains during execution, obligations an actor must obey within a given role or within a set of roles.

We expect that future edutainment WIS will inherit a number of these features and neatly integrate them into the storyboard, into the content management of the WIS, and into the support for activities of actors.

6.4 Identity and Personal Presentation

Identity and personal presentation websites are either identity sites of a company, corporation, or any other business, or oriented towards product presentation, or private sites of somebody. Commercial sites are oriented towards a company presentation, are extended by tours and product presentations. At least the profile and the business is presented. Their main message is “this company – your partner”. Product presentation websites tell the history, the development, and the application of the product. They may be enhanced by user stories, tips, tricks, and news. The vast majority of websites are still private sites. They follow a who-is-who representation of the personality. They may have regions for guests, for partners, for the family, for confession, for leisure and so on.

6.4.1 Branding

In identity and personal WIS the pattern $\mathcal{P}^W\mathcal{U}^A$ is mainly ruled by the first dimension, which is based on a clear understanding of aims and goals of the site including a specification of long-range and short-term goals, anticipated site users, characteristics of the audience, tasks performed by the users, necessary and unnecessary contents, and an understanding of financial restrictions.

For a derivation of goals of the site we consider the “mission” of the organization, figure out which kind of web site would support this profile, harmonise our understanding with the corporate identity, order goals (short-term, long-term), and anticipate an understanding how the site will be in two years from now.

The pattern $\mathcal{P}^W\mathcal{U}^A$ can be specialised for the company identity. The specification of a *corporate identity* is one of the most difficult tasks during website development. Most companies are not able to specify their specific corporate identity. They need however to provide a description of their identicalness, their oneness, and their sameness in all essential characters and in all that constitutes the objective. The corporate identity emphasises that the whole of anything is greater than its parts. A synonym is the word ‘Gestalt’ that encompasses the qualities of form, meaning, and value. It is often helpful to ask for a specification of their likeness, their resemblance, their semblance, their similarity to other competitors, and their similitude. This correspondence, equality, or equivalence provides an insight into the understanding of the company uniformity. The same approach can be applied to personal WIS. We need to get a description of the personality, the selfdom, the selfhood, the selfness, or the singularity of the person. It is often helpful to discuss the distinguishing character or personality of an individual, i.e., their individuality.

[Figure 2.7](#) provides a first description of the psychology of the relationship to the receiver, of the message that we want to deliver to the receiver, and of the understanding of the sender itself. In the next step we can discover dimensions of the communication between the provider and receiver of the WIS. These dimensions are similar to those we used in [Figure 5.2](#): specific service of the provider, specific experience of the provider, content to be transferred to the receiver, service requested by the receiver, background experience of the receiver, and appeal to the receiver. These dimensions are closely related to the facets of intention: aim, objective, target, object, time, and representation.

Corporate Identity

Companies have a broad variety of interest and corporate identity. Personalized business is based on the metaphor of the business men. The variety of business men can be separated into a number of typical behavioral patterns. We distinguish between:

Suppliers (B_S) are companies or people which are acting with goods that concentrate on the competencies of suppliers, their logistics, their services, and their specific key activities such as customer service, demand forecasting and planning, inventory management, logistics communication, material handling, order processing, packaging, parts and services support, procurement, return goods handling, reverse logistics, traffic and transportation, and warehousing and storage.

Providers (B_P) are companies or people which are acting with goods but concentrate on the products they are providing, on their service for the products, their advantages and benefits, and the providers' competence and competition advantages.

Wholesalers (B_W) are merchant middleman who sells chiefly goods to retailers, other merchants, or industrial, institutional, and commercial users mainly for resale or business use. The corporate identity is less complex than the one of the supplier and can be seen as a part of that. In this case the customer-retailer relationship is based on the mediator task of the wholesaler. It can be the basis for the corporate identity.

Wholesalers can be classified into one of three groups: merchant wholesalers, brokers and agents, and manufacturers' and retailers' branches and offices. Merchant wholesalers (B_M), also known as jobbers (B_J), distributors (B_J), or supply houses (B_J), are independently owned and operated organizations that acquire title ownership of the goods that they handle. There are two types of merchant wholesalers: full-service and limited-service. Dealers (B_D) are merchants with a specific style of business. Brokers (B_B) or traffickers are acting as mediators between parties. Retailers (B_R) or shopkeepers are selling goods in small quantities to customers for their own use. Retailing includes all activities required to sell directly to consumers for their personal, nonbusiness use. In this case the customer-retailer relationship is based on the mediator task of the retailer. It can be the basis for the corporate identity. Agent middlemen (B_R) do not take legal ownership of the goods they sell nor do they generally take physical possession of them. The three principal types of agent middlemen are manufacturers' agents, selling agents, and purchasing agents.

Traders (B_T) are selling and buying goods. It can be a physical person whose business is buying and selling or barter. Traders are sometimes acting as merchants who buy and sell (as stocks or commodities futures) in search of short-term profits. Their corporate identity is similar to the wholesaler.

Governmental organisations (A_G) are characterized by their important functions. For instance, city government must provide law enforcement, education, infrastructure supply, regulation of building safety and housing standards, various welfare and health services for the needy and environmental services. Cities also provide museums, parks, playgrounds, and other cultural and recreational facilities. Therefore, their corporate identity is mainly governed by the kind of service they are providing.

Communities and institutions (A_I) represent their understanding of their business, their services, their way of working. This presentation is similar to leaflets that are used for print-media.

Universities (A_U), colleges, schools, etc. are competing for students, funds, and support. Their corporate identity is thus changing from providing a service to attracting customers, i.e., students and fund issuers. Due to the world-wide competition these institutions have also to place their specific competencies into the world market of education.

Individuals and humans (H) use personal websites for presentation of their personality, their skills, their experience, their knowledge, their episode, their adventures, etc. This presentation is seldom based on WIS. Although the website may become rather complex and contain hundreds of pages database technology is not necessary. The identity, self-determination and personality of the individual rules the kind of the website.

The type of the provider directly dictates the content to be displayed, the functionality required, and the scenarios that are of interest within the identity website. The category and the behavioral pattern must be reflected within the entry page. So the design of the entry page becomes one of the crucial activities within identity WISs.

The content, structure and functionality of the identity WIS is often intensively evolving. Society faces nowadays a number of challenges and uses WISs for their resolution. Identity WISs are one of the most evolving systems. For this reason, identity WISs are a good playground for the development of evolvable architectures with a large variety of possible refinements of content chunks, of functionality, of scenario, and of context.

Communication Profile

Identity services heavily depend on the communication profile. In [Figure 2.7](#) four dimensions of understanding messages have been introduced. Receivers or users of identity WIS may often be classified by their specific interest, their tasks, the technology available to them, and their social and organizational context. Based on this characterization, we may concentrate on specific content, on the depth and breath of coverage necessary to engage them, and to provide the functionality so that users can use the WIS. Content may be up-to-date, x-month-old, or historical. Since we are interested in long-term value of WIS historical content is going to be supported. Any link to a source outside the WIS must be maintained.

Identity WISs present the provider and tell how the provider would like to be understood. For this reason, the WIS utilisation portfolio is dependent on the *corporate identity* of the WIS. The task to find an appropriate mission and corporate identity is considered to be a very difficult. At the same time, the utilisation portfolio may also be mistaken since some of the necessary intentions are not taken into account. [Figure 5.2](#) refines the four dimensions of understanding messages to the communication act. It cuts the content to be given by the WIS to content chunks.

The user-provider relationship of an identity WIS dictates the quality of service we are supporting. The relationship depends on the information demand of users. Due to economy of development we are grouping users to actors depending on their information consumption. Based on the relationship we enhance the demand by data users do not know but should know according to the kind of relationship.

Specification of Identity

The determination of corporate identities is often given throughout brainstorming sessions. The most visionary and imaginative people of the company are invited for articulation of the values of the company. The resulting statements are then discussed with the marketing and customer relations departments. This discussion may lead to the description of the unique offer a company has. This description may be afterwards discussed with the heads of the company. It turns often out that differences in their understanding and in discovered values must be resolved. This process of discovering the corporate identity is one choice. There are also other approaches.

Example 6.5. The elicitation of the corporate identity of a hospital for instance can be based on the expectations a patient has. We can ask patients to state traits or qualities they consider positive for this type of company. These traits can be directly reflected by the tone the WIS has. The annual reports are another source of data for assembling the corporate identity. They are usually oriented towards an anonymous audience. One of the best ways for developing the corporate identity is the analysis of competitors with a test team. This test team walks over the presentations of the competitors and extracts their corporate identity by elicitation what they think about this company and what creates a positive or negative impression.

Identity and personal WISs reflect often complex characters. Since users may be interested only in one of the perspectives for the content we can use navigation and search functionality for structuring the utilization. Actors may lookup first for topics by categories they are familiar with, by keywords, or by historical or chronological reference. Typical lookup on the basis of perspectives has been implemented for [GoogleEarth](#). Perspectives may be redundant and overlap. The theory and technology of media types and container specification supports this mosaic-kind of presentation.

Example 6.6. University or institute websites often serve as illustration examples in books and papers. Their utilisation portfolio is however often mistaken or too much narrowed to the needs of their members or even their developers. For instance, students are often hired to be the website developers. For this reason, their needs are often well-represented. Members of institutes do not like to cope with the institute's website too much. For this reason, their personal representation is often somehow given. The utilisation portfolio is however far broader. At the same time the corporate identity changes from being a service provider to the existence of a commercial partner.

Actors of university sites have a specific information need. They range from pupils who want to know which university is the most appropriate, students currently studying at the university, students considering to move to this university, employees of the university, inhabitants from the region of the university, colleagues interested in research and teaching at this university,

governmental institutions interested in controlling the university, and industrial corporations interested in the competency of the university staff. Let us analyse how their information need and information demand actually met in university sites. We consider a number of scenarios:

- Colleague visits colleague: A colleague might be interested in the research and the teaching of another colleague currently working at the given university. He/she is interested in information exchange, new results, projects, news, contacts, and team data. It surprises how difficult it is often to collect all this information.

In most cases, the basket ruled by the actual information demand can only be filled by moving along personal websites. Some universities already started in providing one or two baskets for their employees. Dynamic basket configuration is still missing.

- Student visits institute/university: The student is interested in study information, and conditions, courses, schedules, hand-outs, documents, and in supporting societies such as the student's society. This content can easily be structured into direct course content, general program content, content for a specific course, associations, and facilities the university is providing. The structure is relatively stable. The information demand as well.

It surprises that most of this content is easily available in most university websites. The reason for this 100 % satisfaction is explainable by the development history of websites at universities. In most cases, main developers and programmers are the students themselves. Sometimes content is very difficult to find or hidden so that only insiders may find the content, e.g., course hand-outs. A consistent protection policy for intellectual property and property infringement is missing.

- Staff member wants service: Staff members are interested in a survey of available services, in a search interface for basic staff data, in a search interface of almost all facets of university life, in easy participation in the administrative issues, in tracking content that has recently been updated at the site, in quick survey of news depending on their profile, in support for different tasks they face in daily university life, and in issues of university management.

This content is only partially supported by current websites. It requires consistent updates, dynamic adaptation to the user, dynamic adaptation to the context of the staff member, and dynamic adaptation of the scenario. A typical example of unsatisfied information demands is a surveying facility of Master's and PhD theses currently under development or already completed.

- Pupils or inhabitants visit the university: They are interested in (general) study information, course of studies, conditions for studying, fun, edutainment, useful information for homework, talks, university life, and continued education. Inhabitants might be interested in lectures and evening events they could attend and easily understand. Pupils might consider

study choices and wish to know what is the specific corporate identity of the program within their interest. They need help for comparison and feedback elements.

It is not surprising that most of this information is entirely missing and the scenario is not well supported.

- Industry or information broker visit university: These actors seek for knowledge, experience, projects, seminars, talks, matured valid information, projects, cooperation, contacts, service provided by the university, and continued education. They are not able to nicely integrate their business language with the language used in the scientific society.

Although these actors become more and more important for the university their specific scenarios are rarely met in any university site.

This example shows that utilisation portfolio development is often not taken too serious or is often neglected in website projects. It shows that identity WISs still lack in most content, functionality, and service. The corporate identity of universities looks still the same as it was 20 years ago.

Moreover, metaphors used are mainly based on print media. A useful metaphor could be the presentation as a building. Faculties and organisational units have their floors. Each floor can have its own view to the rest of the university. Another more elaborate metaphor we might use is that one of a crystal ball. Moving from the university in general to its units is then rotating the ball where the rest of the ball remain visible but is zoomed in a different way.

6.4.2 Word Fields and Scenarios

The scenarios we capture for identity services are rather simple. They are based on the corporate identity, on the user demands, on the actor profiles, and on the content we identified. The scenario design becomes crucial since we cannot be interested in random walks through the WIS. During the walk the user needs guidance. In the case of a company WIS scenarios are developed by the customer relations department and by the marketing department.

General guided walk scenarios imitate an exploration through the entire WIS.

All aspects that can be important are shown to the visitor. Shortcuts allow to move to another aspect within the content. These scenarios are entirely governed by the content provided and by the corporate identity.

Most important properties of the provider are shown first.

Viewpoint-oriented scenarios orientate on a detailed survey of one aspect that might be important to the actor. Specific viewpoint-oriented scenarios help to communicate specific qualities to the visitors. We group visitors to actors in such a way that each actor represents one viewpoint on the WIS content. The profile and the portfolio of the actor are matched to content available. This content suite is then harmonised with the corporate

identity. This integration provides a basis for ordering the content suite to content nets with one entry point.

The contact-request scenario is one of the first scenarios to be provided. This scenario also requires a human response channel with the corresponding logistics support in the company for quick answering.

Browsing scenarios are based on the entire content and dependencies among the content. They are supported by a sophisticated navigation support for the user. If the corporate identity has more than one facet then we can separately show corresponding facets depending on the content currently shown. Since random browsing can create a wrong impression we need to create such browsing scenario that browsing through the site has its extra value and significance.

The scenarios can be extended by participation scenarios (^{attend}) and interaction scenarios (^{interact}) with the visitors. This extension is typical for personal identity sites ($P^I 2 V^{attend, interact}$).

Example 6.7. A large percentage of visitors of a university site like the one in Example 6.6 are applicants for admission or visitors thinking of becoming applicants. These visitors usually have a specific information demand. Since they are also known from real life we may develop a number of specific viewpoint-oriented scenarios for these visitors. Some universities provide internet presentations where the applicant has to find out him- or herself where the content can be found, which content is necessary and which restrictions apply. They create thus a rather negative experience for decision making.

Information Portfolio

Users visiting an identity site want to know more about the provider of the WIS. They may have already a preliminary knowledge and may follow certain information demands. A typical information demand is caused if somebody applies for a job at a company or prepares for an interview at that company. Another actor is interested in rating the company, e.g., for commercial issues such as contracting, considering partnership, or within the portfolio of financing institutions.

Example 6.8. The information demand and the content necessary can easily be detected if we use the characterisation of actors. Let us consider a fund-raising company such as www.rotary.com and their actors. Financial sponsors may ask for reasons why to sponsor, what is going to be done with the donation, who are the beneficiaries, and how to make a donation. This information demand is satisfied by the data about the company, statistical data about the usage of donations, overviews of beneficiaries and their stories, and instructions for donation making. Volunteers may ask for information why their participation is worthy, what volunteers usually do, and how to get involved. Staff members are interested in volunteers, beneficiaries, and financiers. The

other actors are additionally interested in reports of the latest developments and on projects. This information demand is satisfied with message boards for current discussions, headlines and blurbs for display of new events.

Identity WIS should provide basic data on the provider: who they are, what they do, and why we should trust them. Therefore, the company WIS must provide information on the organisational profile, on contact information, a disclaimer and legal information, on customers and partners, on employment opportunities, on public relations, on investor relations, on community relations, site credits, and for frequently asked questions. The WIS should have a link to the privacy policy. A good feature of the ‘page-not-found’ error page is a link to contact information.

Main content of identity WIS is the introduction to the company or personality, to the partners, to the products, and to the people. This content is extended by addresses, locations, and references. Based on the intention the know-how, technology, cooperation and interest, contact, passion, abilities, and profession are described. Companies may also present their profile and provide life stories about their business.

Action Verb Fields

The corporate identity is not only given through the atmosphere of the WIS but should be supported by content. This content contains a clear and persuasive promise and a unique offering. The content is given through descriptive wording and images that are easily and quickly understood.

We identify five main word fields for this category of WIS:

Introduce: The company, corporation, or person introduces themselves to the visitor. The introduction has its own value for the visitor. The introduction is based on a presentation, premises, or preliminary explanatory or laudatory remarks.

Inform: The presenter provides content of special interest or importance. Information presentation implies sending notice of content requiring attention or demanding action. It imparts knowledge of some fact, state or affairs, or event to the visitor. We already analysed this word field in Chapter 2.2.

Show: The word field *show* satisfies another intention of the WIS. The WIS is set out or placed on view for potential customers or collaborators. The competency of the provider is made apparent and revealed. The provider is presented in such a way as to invite notice, attention, and admiration of the visitor. The main actions of the visitor are specified by the brand pattern ($P^I 2 V^{\text{look, come_back, make_out}}$).

Present: The word field is less ambitious than the previous two word fields. Content is demonstrated and delivered to an interested audience. This content can be taken as granted. The provider is portrayed. The general brand pattern is then ($P^I 2 V^{\text{look}}$).

Know: This word field describes the most ambitious intention. The visitor should know the provider, the competencies and potential, the ambitions, and the content. This word field requires powerful adaptation facilities. The WIS is similar to learning systems. The visitor acknowledges, recognizes, and accepts what is claimed by the provider.

These different word fields can be used to develop the intent and purpose of the WIS. They result in requirements to content. Depending on the scenario chosen for the WIS we may directly derive the form, the content and the main functionality provided by the WIS.

6.4.3 Adaptation

A special feature is guidance for the visitor and self-adaptation of the site to the visitor. Guidance helps the visitor in navigation throughout the WIS, in supporting understanding the content provided by the WIS, and in context-sensitive help whenever a visitor cannot complete the task under consideration. There are a number of features that are easy to implement and provide the guidance a visitor might need: examples of what the visitor can or should do, optional meaningful messages for each action a visitor might use, on-demand detailed description of all features provided by the WIS, step-wise explanations, and shortcuts to frequently asked questions. For first-timers we can provide tour guides that tell the story or the stories of the WIS. Guides provide value for visitors even if they don't pursue links and navigation paths. They can be customised to different actors and information demands. Guides are particularly important for WIS with restricted access for visitors in general because they demonstrate the additional value if the visitor becomes a customer.

Adaptation to visitors is a must if optimal visitor support is aimed at. The WIS realizes a limited visitor adaptation in that it offers in the sub-scenario selector for different categories of visitors. The self-adaptation facility consists in identifying the subspace of the story space they create that most likely will fit best the information demand of a particular visitor. The match between visitor and subspace can be done such that the visitor is asked to give some characteristics of him or her into the system and based on that the respective subspace is chosen. This strategy is suggested by the fact that the provider in general does not know much about the individuals accessing its site. A technique to consider knowledge about customers to the design process is the creation and use of personas, i.e., archetypical customers, and design of the navigation structure as well as the page layout such that it fits optimally to the personas used. Based on an automatic visitor assessment that in response to his or her site-interaction updates the scores in each of the user dimensions throughout visitor-site interaction one can then track how a visitor's trace moves through this space and detect when a modified type would better fit the customer's behavior than the actual type does. Clearly such update should only be done with visitor permission.

Typical functionality limitations of print media apply to WIS as well. Content may be restricted for export. Due to bandwidth we should provide content and functionality with self-adaptation or at least user-configuration facilities. This kind of tailoring can be extended to singleton-person presentation. Tailoring should be limited since we will not be interested in ‘Balkanisation’ of content provided that leads to splintering off the set of potential users into tiny communities that cannot interact with one other.

Quality requirements to identity and personal presentation website differ from most other categories. We aim in providing a true picture of the competency or corporate identity of the company or the person. So, quality criteria to be considered are: accessibility, branding, community, consistency, ease of use, feedback, flexibility, guidance, immersion, preview, responsibility, responsiveness, simulation, speed, storytelling, style, support, tone, and understandability.

6.5 Learning and Edutainment

A large number of WISs provide learning and edutainment services (see also the discussion in [742]). Examples of technology-supported learning include computer-based training systems, interactive learning environments, intelligent computer-aided instruction systems, distance learning systems, and collaborative learning environments. The early enthusiasm has decreased, and developers have learned a number of lessons:

- Edutainment should neither be considered to be yet another presentation form for classical instructional learning nor a means for the presentation of content that has been used in lectures.
- Edutainment should not be understood as the enrichment of learning by multimedia facilities but must concentrate on content that is easy to use and to understand, is enriched by functionality that corresponds to the content, permits controlling the progress in learning, and is presented in a pleasing form.
- Edutainment can support anybody willing to participate in learning activities at any place any time within any team. However, learning cultures will limit this claim, as learners have their own profiles and portfolios.
- Main kinds of learning stories are complementary learning, self-organised learning, and continuous education on demand. Learners must therefore be supported in developing their own learning space and in understanding and developing their learning abilities.
- Control and assessment cannot be completely automatised, as individual learners, contexts and learning history differ, and learners often need immediate feedback.
- Edutainment can be supported by a number of devices ranging from computers to PDAs and mobile phones. We thus distinguish between electronic

learning (e-learning) systems and mobile learning (m-learning) systems. Mobility in general is changing people's way of working, communication and learning, and consequently, the differences between formal and informal learning will diminish. The real added value of m-learning is in the area of informal learning, whereas e-learning may cover formal and informal learning.

- Scenarios and content are multi-faceted depending on the presentation device, and must be adaptable to the learner and the learning community. The style of learning changes to proactive formal learning with self-study content, virtual classrooms and trainer-based facilitation. Learning becomes social, arguing, reflecting, articulating, and debating with others.
- The vast majority of learning on the web is informal, based on ad-hoc information sharing, communication and collaboration. Thus, the functionality of edutainment WISs is also based on functions such as participation, contribution, and annotation.

6.5.1 Branding

The general brand $\mathcal{P}^W \mathcal{U}^A$ of a WIS has to be specialised for edutainment WISs:

Provider \mathcal{P} : For the time being providers are mainly educational institutions or educational communities. Then the provider plays the role of a teacher.

Product \mathcal{W} : Since control and assessment of learning progress is a still unsolved issue and appropriate presentation of complex information is often not feasible, edutainment WISs concentrate on easy-to-understand information or easy-to-grasp knowledge.

Users \mathcal{U} : Users of edutainment WISs are mainly students, people seeking continuing education, workers in companies with a specific portfolio, people interested in auxiliary information, or groups of such users. The main behavior of such users is characterised by the role of the learner.

Activities \mathcal{A} : Activities are currently centered around learning, searching for content, collecting content, and solving exercises. Activities also include asking questions, acting in teams for problem solving, and discussing issues associated with the learning material.

Thus, typical learning WIS brands take the form **Teacher $^W \mathbf{2Learner}^A$** , where \mathcal{W} stands for content_chunks or knowledge, and \mathcal{A} can be one of {receive, respond, solve_in_teams, raise_questions, possibly_apply}, {learn, validate, control, advice}, {discuss, get_feedback, work_on_it}, {recognise, listen, work_on_it, solve_exercises, ask_questions}, etc.

Edutainment WISs are currently mainly or exclusively supporting student actors. Students obtain knowledge through teachers, their schedules, and their abilities; they need guidance, motivation, and control. The behaviour of actors may, however, be more complex, in particular in case of learning groups,

in which the collaboration of students depends on a cooperation profile with rights and obligations. Communication partners exchange content, discuss and resolve questions, seek hints or better understanding, supporting and motivating partners are users with control, motivation and supporting functions. Furthermore, teachers have various obligations and rights, and are involved in a variety of roles.

6.5.2 Word Fields and Learning Scenarios

Modelling of e-learning scenarios is more difficult due to the variety of didactic approach, to the variety of learners, to the variety of tasks and the context of the site. For this reason, modelling of dialogue scenes and dialogue steps is more generic than modelling of simple interaction steps that usually lead to deterministic runs through the story space. We, thus, distinguish a number of associations among steps and scenes depending on the content of the learning element.

Our solution to this challenge is based on generic parameters that are instantiated depending on the learner, the history, the context, etc. Each learning unit is specified by a context-free expression with a set of parameters. These parameters are instantiated depending on the learner profile, the learner task portfolio, the media objects, the learner computational environment, the data to be used in exercises or algorithms, the presentation environment, and the available and accessible learning elements.

Learning scenarios (^{learn}) are classically based on general learning styles:

Sequenced learning is based on a curriculum sequencing in active or passive scenarios based on classical pedagogical approaches. Typical established pedagogical approaches are conditioning, operated learning, model learning, cognitive approaches. Sequencing is extensively studied based on classical didactics. Active scenarios model behaviour of active actors. A typical metaphor to be applied is the shopping bag. Passive scenarios did not get the success that has been expected during the 1980s and 1990s. Tutoring systems are applicable for advisory systems, for help desks, or for training physical skills.

Interactive learning either in self-organized or content-sequenced or habit-regulated or publish-subscribe scenarios.

Group learning in a cooperative setting (integrated sub-tasks, black boarding) or in a collaborative setting (cooperation, discussion, development of solution by all members).

Sequenced learning is currently mainly based on didactic approaches, i.e., on instructions, classical content and classical hermeneutics. Due to the sequential way dependencies among learning units are rather simple. Learning scenes use reception techniques, explication of the learning content, context association, deriving understanding and interpretation, and finally integrating the essence of the content into the knowledge of the learner.

Learning Content

Content used in edutainment WIS is mainly easy-to-grasp and easy-to-understand information or knowledge. Most important associated scenarios are validate, control and advice. Content is given in a large variety. We need therefore to consider the kind of content, the activities that can be associated with the content, the characterisation and annotation of content, and finally the quality characteristics of content. Edutainment content delivery, storage, retrieval, and extraction is still an open research issue.

Edutainment WISs provide content to learners depending on their learning task, their personality, their working environment, their learning history and the policy of the content provider. This challenge is more complex than the challenge to generate “content”. Content can be represented by media objects. Content for learning must have high adaptivity. We distinguish a variety of content or generally knowledge:

- knowledge and abilities for *orientation*, e.g., explanation, presentation, history, facts, surveys, overview;
- knowledge and abilities for *application*, *skills*, *abilities* rules, procedures, principles, strategy, and laws;
- knowledge and abilities for *explanation*, e.g., ‘why’-knowledge (proof, causal) and ‘what’-knowledge (definition, description, case, argument, assumption, reflection);
- knowledge and abilities on *sources*, e.g., archives, documents, citation, reference, and links;
- knowledge and abilities for *solving problems*, e.g., sample solutions, analogs, training solutions, discovery solution, and examination.

All media objects can be provided independently of the learner. We need however to consider also differences:

- *Background knowledge* leads to different speed and reception by the learner.
- *Work abilities and habits* influence current work.
- The *learning style* must be considered in many facets.
- The *social environment* is based on cultural and psychological differences.
- The *history of the learning process* should be considered if we want to avoid annoying repetitions.
- The *learning portfolio* influences occasion, intention and motivation.
- The *learning object presentation* is or is not acceptable depending on the profile of the learner.
- The *learning environment* is modelled by many technical facets.
- *Content change management* allows to provide content with or without refresh.
- The *payment profile* may result in content reduction.

Learning objects are elements of a new type of computer-based instruction influenced by the object-oriented paradigm of computer science. Learning objects are defined here as any media object that can be used, reused or referenced during technology supported learning. Examples of learning objects include multimedia content, instructional content, learning objectives, instructional software and software tools, and persons, organizations, or events referenced during technology supported learning. The IEEE's Learning Technology Standards Committee has developed an internationally recognized definition of "learning object": "any entity, digital or non-digital, that can be used, reused, or referenced during technology supported learning". This definition is extraordinarily broad.

A large variety of learning elements is used in edutainment WIS. Course elements are lecture notes and comments. Exercise material may be given in a textual or in an interactive form. Illustration material is often based on animations, complex multimedia elements or on links. Algorithms can be provided in an executable form, as virtual machine or via an interface to the server. Input data for algorithms can be provided with the learning elements or through other elements in the network. In the DaMiT chapter of this book we show how to properly extend this facility of learning objects. We shall use learning elements, learning units, and learning modules. All these types are media types or container types.

The use of *standardized metadata* to assist the content adaptation process is quite central to the distribution of content to diverse and heterogeneous environments regardless of the coding format. Metadata describe the usage environment, which includes terminal capabilities, network characteristics, user characteristics, as well as characteristics of the natural environment. These tools will steer the adaptation process to output content suitable for delivery and consumption within the given usage environment.

Edutainment WISs need to supply also a support for context integration. We distinguish between

- context for media types on the basis of the media type suite and associated suites,
- context through association of specific content,
- context provided by the story space associating stories and media type suites,
- context due to the frame, intention of the website that integrates also content not associated to the current content, and
- context given by pragmatical instantiations due to the WIS utilisation space and to the user model.

Action Verb Fields

In order to derive the utilisation portfolio we analyse the word fields associated with common verbs in the learning context such as learn, know, master, study

and nouns such as skill. This gives rise to stories and consequently determines the functionality requirements.

Learn: Learning is a very complex activity that includes gaining knowledge, understanding, obtaining skills, etc. by studying, instruction or experience. In addition, learning is associated with memorizing, being able to perform some task, and to know this ability. Learning is based on obtaining content, discovering the concepts behind them, annotation, ordering, and integration. Learning is associated with the role of a learner or student, who are usually supported by other actors who teach and instruct. Learners determine content with certainty, usually by making an inquiry or other effort. They check the content to find out whether it is useful or whether additional content is needed.

Know: As learning aims at improving skills, abilities, and knowledge, the improvement should be measurable and learning success examined. Knowing means to be cognizant of a fact or a specific piece of information and to possess knowledge or information about it. It may also include knowing how to do something. Learners obtain first-hand knowledge of states, situations, emotions, or sensations, and the change in knowledge is acknowledged and recognized by other actors.

Master: Learning usually intends to enable mastering problems and to become proficient and skilled in some area. This mastership is closely related to practising and experimenting with the new knowledge.

Study: Studying refers to the activities associated with learning, i.e., reading learning content, excercising, checking, examining, inspecting, surveying, etc. Therefore, the presentation of the learning material and the storyboard are essential. Furthermore, learners need to mind, perpend, think (out or over), and weigh, which requires time and workplaces. Studying can be performed by oneself without any teacher, supporter, or observer.

Skill: Skills are abilities that have been acquired by training, e.g., abilities to produce solutions to some problems. A learner is trained until he or she obtains these skills.

These word fields have a complex structure, which leads to functions requiring other support functions associated with related word fields such as discover, ascertain, catch on, and determine. One difference with the word fields for e-business is their iterative application. The word fields are extended by the learning style to be supported. The three different learning styles – sequenced or blended learning, interactive learning, and group learning – result in rather different scenarios. So far, only sequenced learning approaches have received the necessary theoretical basis in pedagogical research and didactics. Interactive learning is still an open research issue.

Example 6.9. The KOUPRA project within the German University Notebook Initiative was aiming at the support of adaptive and collaborative learning,

focusing on practical training in database programming. The system is supporting communication and interaction of learners, organization and coordination of collaborating communities, free access to material depending on the progress, roles and portfolios of partners, and the stepwise development of training material. Learners act in a collaborative setting depending on their needs and the goals of the learning program.

The analysis of the word fields above highlighted the utilisation of importance of exercising of crucial part of the WIS. While often only multiple-choice exercises are supported, complex exercises have a more complex pattern. [Figure 6.3](#) shows the derived scenario for exercise training for this case. Learners are allowed to choose either their own data or data provided by the system. Learners may also choose an algorithm that can be used for solving the exercise.

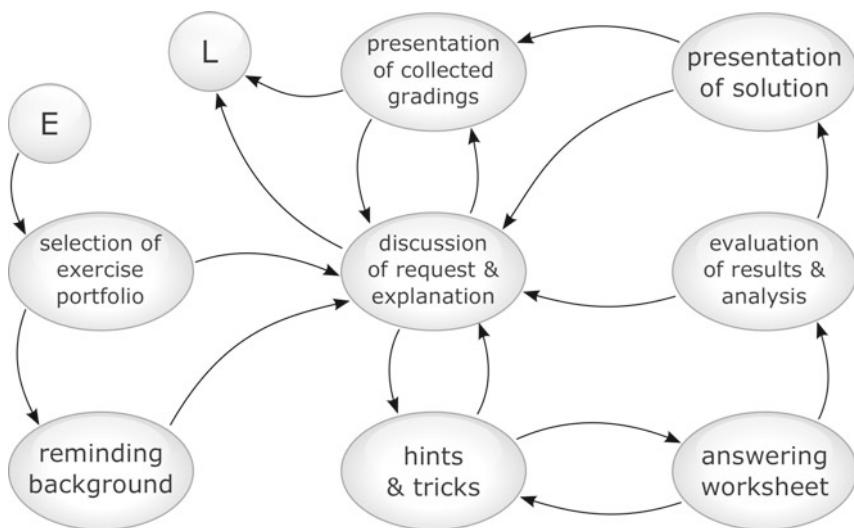


Fig. 6.3. KoPra scenario for training with optional exercises

Using the storyboarding language **SiteLang** we can represent the entire scenario by the following expression:

$$\begin{aligned} &\nearrow T; ((D \square (C; P)) \parallel (I; U)); \\ &H; (R; H;)^*; A; ((S \square \text{skip}); E; J; N; H; (R; H;)^*; A)^+; S \searrow \end{aligned}$$

with the complex actions T (task delivery), D (delivery of prepared data), C (collection of users data), I (information on applicable algorithms), P , (preparation of learners data), U (code upload and installation), H (formulation of learners hypotheses), A (computation of association through mining),

S (submission of competitive solution), E (evaluation of submitted solution), J (inspection of sample solution and comparison with evaluations of competitors), N (preparation for next trial for solution), and R (reminder on learning element on hypothesis). Further, the symbols \nearrow and \searrow are used for denoting the entry and the termination of the whole exercise scene.

6.5.3 Supporting Features

Learning content must be properly handled. It turns out that this task is one of the most difficult tasks. For this reason, any edutainment WIS must be combined with an *authoring WIS* that supports authors during appropriate development of content.

Quality Control

Quality control of learning objects is necessary since low quality data harms the learning success. We may distinguish a number of reasons for low quality and development strategies for improvement of quality.

- *Incomplete content* is caused since data are partially known or partially missing.
- *Mixed content* appears whenever learning objects come from various sources. We need specific integration, versions, expansion and extension, and timeliness features in the system.
- *Wrong content* is caused by various sources of errors, from computational to observational to interpretation errors.
- *Complex associations among content* may lead to confusion or chaotic scenarios. We need thus a proper decomposition of content into almost independent modules.
- *Mutated content* is often caused by partial minor changes within the content due to a number of reasons.

System support should provide a flexible adaption to different roles of users and different content. In high abstraction, the learning space can be characterised by the three dimensions depicted in [Figure 6.4](#): The learner's and teacher's dimensions abstract from the concrete learner and teacher and groups learners and teachers by their general characteristics. The content dimension characterises content in dependence on the support that is necessary for its delivery.

Classification of Edutainment Systems

Most edutainment systems are currently based on scenarios of sequenced learning. This generation of systems will remain but will also be superseded by systems that provide facilities for interactive or collaborative learning. The

Intelligent Content Support

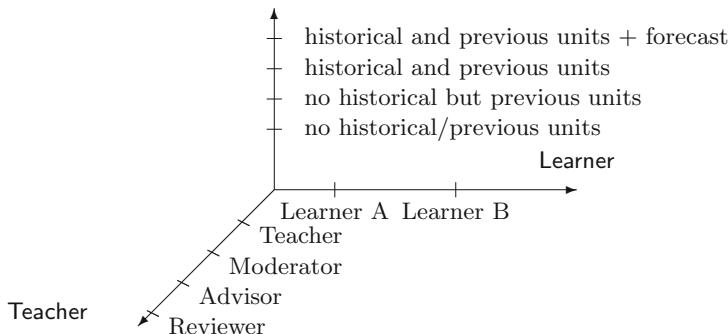


Fig. 6.4. Three dimensions of the learning space: content characteristics, learner, and teacher

advantages of WISs will be only visible for the last two categories. Sequenced edutainment WISs are far better embedded into classical technologies such as print media and CD-ROM media. We thus may distinguish between three generations of edutainment WIS:

- 1st generation systems of the 1980s and 1990s did not succeed since they were demoralising users with tutoring systems in a strongly sequenced curriculum.
- Re-engineered 1st generation systems of the internet age copied classical teaching approaches and have been mainly content providers. In most cases they allowed to put your slides onto the web and to advertise the course. These systems provided a minimum of support, mainly for control of the learning success. Most workbenches currently established are still supporting this kind of teaching and learning.
- 2nd generation systems provide learning objects with customizations, with adaptation and with content management. These systems support adaptation to the user, to the learning progress, and to the learning history. These systems remained mainly to be prototypes due to the difficulty of content development.
- 3rd generation systems are aiming in providing best-suited content just in time to the right user, place and device with the best pricing. These systems will be the main systems of edutainment that support sequenced learning. They are currently mainly developed in university research projects and far from being mature for application with larger learner communities.

There are almost no systems available that support interactive or collaborative learning. The reason for this lack is mainly the poorly developed pedagogical basis. Didactics is still mainly oriented towards classroom teaching.

3^{rd} generation learning systems provide a number of features and challenges:

- Learning support is far more complex. Research is sought on didactics, content integration and delivery, storyboarding, adaptation and context integration, and success control.
- Open learning objects provide a sophisticated facility for content management. The theory of open learning units should be integrated with didactics based on storyboarding, content adaptation and delivery, and content development. In future, they will be extended with context, e.g., story space, actor, user, payment, portfolio, association, history, etc.
- Control functionality should be provided for open learning units in the same fashion as we know it already for exercises, tests, and exams for self-control or certification.

6.6 Information Services and Infotainment

Information services aim in delivery of information to customers depending on their information needs. Typical providers are government, news companies, institutions, and individuals. The goods delivered are data, news, and messages. The customer can be an individual, government, or everybody. Typical activities are read, become informed, understand, and become attracted. Information scenarios are based on the word field inform or more specifically ask and search . Therefore, we specialise the brand pattern $\mathcal{P}^W\mathcal{U}^A$ to the more specific brand B^T2V^{inform} , A^T2V^{inform} , C^T2V^{inform} .

Within our projects we developed information-intensive sites which need a sophisticated database support and which are based on a variety of various stories of their use. For instance, the $(A,G)^T2(V,C,A,B)^{\text{inform}}$ -scenario is a typical for city information WIS.

Information services are usually based on a broad-band portfolio and are targeted to almost everybody. For this reason they must be robust, flexible, and adaptable.

Example 6.10. The *Cottbus interaktive* project is an information service that provides media data such as TV and radio programmes, electronic program guides (EPG), video streams, video text, and also internet data based on cable nets and set-top boxes. The project showed how different media can be combined into one melted and combined service. TV viewers also obtain video stream on their request, EPG information they need, and internet data depending on their selection. The set-top box has a ruler-based navigation and interaction. Therefore, the viewer must be profiled according to the chosen interest and information need. The profile is adapted during the utilisation history. The user's history is thus recorded and used for lazy adaptation of their internet profiles.

The TV viewer is interested in TV/Radio-on-main-interest, EPG-on-profile, Video-on-request, Cinema-on-demand, Greeting-and-messages-upon-request, Internet-on-interest, and Internet-on-profile. The utilisation space of the information services is therefore also based on profiles and portfolio of interested and informed users. It also reflects the technical environment such as channel capacity, communication protocols, and demands of supporting competitors. Such kind of information challenges current technology. The service that is currently in use is based on powerful servers, powerful delivery channels, thin request channels, and browser-based set-top boxes.

6.6.1 Storyboard Development

We now outline actor specification, scenario specification, content chunks and the derivation of the storyboard.

Actor Specification

Information services are targeted to a broad variety of actors. The main kind of actor is the visitor who becomes informed and attracted depending on his or her information need or demand. Visitors can also be guided and may obtain value-added content. Casual visitors are mainly attracted. Visitors may also be categorised as foreigners, (fe)male, pupils, kids, youngsters. In this case they need specifically adapted content. Pupils, kids, and youngsters often expect to be supported for gambling, etc. similar to entertainment WIS. Some visitors may be handicapped. In this case the presentation must be based on specific pattern and grids.

Another kind of actors are customers. They become attracted after becoming informed. They require functionality such as browsing support and search facilities. Actors may also be more official such as people working for government. They want to become informed depending on their targeted information demand.

Actors may be expert for a certain part of the information service and layman in another part. Adaptation is becoming a challenge for the development of the WIS depending on the personality profile of users.

Actors have a number of reasons for visiting the information service. Typically, the visit is demand-driven. It depends on the information situation of the user, is time-dependent and on the equipment of the user and supported by the WIS. Actors are classified by their profile and typical properties such as age group, handicap, and nationality. The behaviour of actors is classically based search frames. The financial situation of users may also restrict access.

Scenario Specification

Information services support a variety of scenarios. We may classify scenarios into the following kinds:

- Attraction scenarios aim in attracting, informing or guiding visitors. They are extended by special offers and bargains or games.
- Information scenarios aim in informing and guiding the user depending on the information demand of visitors depending on the information level. They are extended by search scenarios depending on the search portfolio and the search abilities of visitors.
- Targeted information demand scenarios provide shortcuts to targets and add facilities for informing, providing, directing or advising.
- Information service scenarios for community or group member are based targeted information demand and additionally combine it with community services or entertainment services.
- Browsing scenarios support the zapping or browsing visitor. They provide content piece-by-piece and they attract or inform.

Example 6.11. City and region information services like the www.cottbus.de WIS typically realise a large number of scenarios. Typical scenarios are, for instance,

- scenarios providing content on the city or region, on the history, etc.,
- scenarios providing content for short-term visitors of the city or region,
- scenarios providing content supporting group visits and group entertainment in the city or region,
- scenarios providing data on restaurants and places to go,
- scenarios providing proposals for combined activities for visitors,
- scenarios providing content on events such as cultural or sport events,
- scenarios informing on companies supporting tourism such as travel agencies or hotels,
- scenarios providing service for travellers travelling through the region or the city,
- scenarios providing service for people stranded in the region or city,
- scenarios providing content for inhabitants, e.g., spare time, official information,
- scenarios supporting people who moved to the city or region,
- scenarios providing background content for all other scenarios,
- scenarios providing data on actual highlight and possibilities for their visitors, and
- emergency information for inhabitants and visitors.

These scenarios already require specific content, specific functionality and specific presentation.

We discuss in detail information services in the last part of this book. For this reason we can restrict the development of utilisation portfolio to these general remarks. Information services must support scenarios which are more related to breadth-first search. Visitors often want to know in advance whether the information they are seeking is really provided. In this case, most

scenarios are elaboration scenarios. They first survey opportunities and later select most appropriate opportunities.

Example 6.12. Self-service governmental WISs are a typical elaborate kind of information services. The inhabitants' portal built within the website project www.cottbus.de provides support for a number of life cases. A typical life case is the official registration in the case of relocation in Example 5.8. Inhabitants first explore which steps must be considered, which documents are necessary and which related life cases they can complete as well. After this exploration of the story space they decide which steps they will undertake.

Content Chunks

Information services provide content. The utilisation scenarios are thus content-driven. Content and organisation are the critical success factors for information services. Context must be given explicitly. Privacy is often another success factor. Utilisation depends on the portfolio and profile of the users. We may distinguish a variety of utilisation:

- Intention-based utilisation is driven by the intentions of users. They use the information service for search, for leisure, study, travel, spare time contacts, get together, etc.
- Life-case-based utilisation is based on assembling supporting material. Information services provide support for life cases and guide the visitor within their life case. Typically, users also need answers to frequently asked questions.
- Portfolio-based utilisation extends the previous kinds by a workspace, a workplace, a solution space and/or instruments for the solution of tasks. These information services support the tasks of user by appropriate content. This kind of utilisation is often combined with community services or collaboration facilities.
- Profile-based utilisation follows either the educational or personality profile of the user.

Story Space Derivation

The scenarios are combined into the story space. This combination allows also to derive additional scenarios which are based on combining sub-scenarios with other sub-scenarios.

Example 6.13. The combined story space of the www.cottbus.de WIS uses the fork | into business, culture and tourism, and inhabitant scenarios. These scenarios partially reuse the same content or are based on content that is derived from content. For instance, traffic information for inhabitants is far more detailed than the one for tourists or business people. Inhabitants do not

need a portrait of the city whereas tourists and business people get different portraits which are derived from the city portrait. The story space can be thus given by the expression:

```
Entry {
    Business { Portrait1 | Technology region | Business parks | Business partners
              | Business environment | Development projects | Exhibition, fairs, events1
              | Traffic1 }
    | Culture & Tourism { Portrait | Tourism service center |
                          Calendar of sport and cultural events | Walking & dining | Spare time |
                          Leisure | Exhibition, fairs, events2 | Sport | Traffic2 | Weather }
    | Inhabitant { Political life | Administration | Official, regulations |
                  Inhabitant guide, inhabitant service | Social, associations | Education |
                  Traffic | Exhibition, fairs, events | Emergency | Weather } }
```

This expression combines the scenarios given in Example 6.11.

Information services are providing a large or huge variety of content. The content is extracted from a number of databases. For instance, the www.cottbus.de WIS uses more than a dozen different databases. The content delivered combines data from these databases. Due to this large variety of content we need a structuring facility for the content. Depending on the utilisation we use a variety of content organisation and structuring schemata:

- Intention-based organisation is based on categories of content. This categorisation may vary from country to country and from region to region. Intention-based content organisation can be derived from knowledge categories or from typical information situations.
- Portfolio- or task-based organisation uses the storyboard or inner organisation of the tasks. Content is related to subtasks, to the context, or to conditions for starting or completing tasks.
- Life-case-based organisation combines the content supporting the life case into a content workspace. Different views of this content are provided depending on the stage of the life case and depending on the user.
- Profile-based organisation uses the structuring that is either selected and refined by the user or that is associated with certain properties given in the user's profile.
- Content has also an inner order and an inner logical structure. This structuring schema may vary depending on the information level of the user and on the information demand of the user. We may either use the content-driven organisation as a micro-organisation schema or for linking content to related content.

Most city and region information services use intention-based content organisation. Information services supporting work or leisure should however use a task-based structuring. Information services of companies are often used by maintenance people. In this case, the content structuring is based on the tasks of the service personnel.

Example 6.14. The information service of the Cottbus housing companies provides information for all who rent a flat from those companies. The content organisation schema is mainly driven by the portfolio of the housing service. Additionally, call content is provided for emergency cases, for service personell, for energy and gas companies, for infrastructure service, and for the housing companies. These companies also use services for distance control of their buildings. Some of these companies are association-based. Therefore, the information service is integrating facilities of community services.

The structuring schemata are combined with links or content to other parts of the WIS. The combination may be tight or loose. Loose combination is often based on links or other navigation elements. We will use either media types defined over media types or containers.

Example 6.15. The www.cottbus.de WIS uses a variety of intention-based organisation schemata and a view on the content. For instance, the restaurant guide provides basic data on the restaurants, their menus, their location, their reachability, opening hours, etc. Restaurants are classified according to their style of cooking, their interior, their visitor communities or associations. It is linked with the events guide for those events which are located in these restaurants and for those events which are close to restaurants. Some of the restaurants provide regional cooking, others foreign styles. The first kind is extended with some regional information.

6.6.2 System Organisation

The most difficult problem for content provided in information services is the update problem. The problem becomes more complex if we introduce redundancy due to the vast variety of represented content that is based on the same data. Information provided in information services has a life span.

Example 6.16. The FuElne project was aiming in providing data on companies in Germany for one of the publishing houses. These company data must be as actual as possible. At the same time, the data are assembled into a large variety of views due to the large variety of information requests issued by companies, by the government, by authorities, etc. We used hundreds of indexes for the support of these requests. At the same time, company data often change due to the competition in the market. Some of the changes are not noticed. Therefore, the indexes must also be robust against these missing or wrong values.

The life span of company data can be defined as the time in which the data we collected is actual and valid. We measured together with the publishing house the “half life time” of such data. Within this project we learned that the half life span of these data has been around six months.

The functionality of information services is driven by content, ruled by the storyboard and oriented on the user demands. Since content provided by an information service is often huge, we need sophisticated database technology for the support of the following subsystems:

- The search and retrieval subsystem supports user requests with facilities for targeted or elaborate search and retrieval based on a variety of ordering and structuring schemata. Users want to see content piece by piece or first in the general summarized form. Typical integrated database systems are city maps, address dictionaries and traffic directories.
- The procurement subsystem is a typical logistics system that supports procurement and generation of data, transformation of data into a format that can be stored, and recharging the existing databases of the information service.
- The redaction subsystem allows to modify the data obtained. It supports the redaction team in their data acquisition work, the organisation of the data, the control of data quality, and in restructuring the databases.

Typical actors are the *redactor* who is moderating, the *data supplier* who is distributing, bundling, presenting, and marketing data, the *system provider* who is in charge for the database systems behind the information service, the *channel provider* who supports the service with channel capacity, the *internet provider*, and the *terminal provider* who can also support the service by terminal facilities.

The storyboard often requires specific adaptation of available content. Therefore, the information services must be based on a powerful generation facility such as the theory and the technology of media types. As interaction types can be built based on database types and this construction is iterative we are able to support ‘towers’ of media types similar to ‘towers’ of views that are used in advanced database applications.

6.6.3 Life Cases and Derived Functionality

The storyboard is based on life cases we envision for the application and on scenarios we have modelled.

Life Cases

The life cases and scenarios can be analysed for their basic units. Since both are either specified based on natural language or based on our semi-formal specification templates we can use word fields for representation of basic units. We consider two different kinds of word fields for information services:

- Substantive word fields are mapped to content chunks which are containers for content or media object suites. These word fields result in a large variety of requirements for content.

- Verb word fields are mapped to functionality requirements. These functionality requirements are typically based on a small set of typical word fields.

We can distinguish typically five main word fields for information services:

Inform: The main aim of an information service is to inform the visitor according to his or her information demand and according to the profile and portfolio. Data are received and understood by visitors and reduce the recipient's uncertainty. They are classically a collection of facts from which conclusions may be drawn or knowledge acquired through study or experience or instruction.

Guide and direct: Information services steer and direct the visitor through their service. The visitor is proactive according to the data that is provided. Guiding includes also helping the visitor whenever help is appropriate.

Provide: Information services supply data to the visitor depending on what the visitor desires or needs. Therefore, the service must determine what the visitor needs and demands.

Search: Searching is a very complex word field which we discuss in the next chapter in detail. Visitors look into or over carefully or thoroughly in an effort to find or discover something within their demands or interests. They examine and check the content provided.

Lead: Information services take somebody somewhere. This leadership results in a change of the information stage or level of the visitor. This change must also be taken into consideration during later visits. The leadership may cause one to undertake a certain action.

Information services are often supporting *infotainment* of visitors. Additional word fields we use for the description of scenarios are:

Advise: The information service recommends visitors. They give information or notice to information. The visitor consults the service for his or her information demand.

Attract: The services cause to approach or adhere to the visitor. The visitors need to be fascinated and enchanted and so compelled to a response.

Browse: Information services are consumed by the visitors. Visitors look over casually or through media objects casually especially in search of something of interest.

Offer: Services present data for acceptance or rejection in order to satisfy an information demand. They propose, suggest and make available data.

Weave and network: Content provided by the information service is united in a coherent whole. Visitors produce their own understanding by elaborately combining elements, by intertwining, or by interlacing.

Zapping: Some visitors propel suddenly or speedily through the service. They avoid watching or reading by changing services especially with an address controller or by fast-forwarding an address directory.

Derivation of Functionality

We can use these word fields for derivation of requirements to functionality of the information service:

Information, communication and transaction functions: These functions support visitors in their work, their information demand, and within their intentions. Additionally, all other actors of the system are supported within their portfolio.

Feedback functions: Feedback functions allow the visitor to give a feedback which is going to be answered by the addressee. Therefore, feedback functions are supported by a logistics system for answering and reacting.

Import and export functions: Visitors often want to print or integrate the data they got into their personal workspace. Export functions are a must for any information service. Some information services also support business or community scenarios. Therefore, import of data must be supported in a very flexible way.

Combination functions: Users often want to combine the information they obtain without opening a number of additional browser windows. They can be supported by combination functions whenever data is carried to a basket or to the user's workplace or workspace. Typical life cases already require such functionality.

Orientation functions: Due to the large variety of data visitors need a guidance and an orientation facility beyond navigation bars. Orientation functions allow the visitor to track where they are, how they came, what they can do, and which path they can use for their demand.

Search and retrieval functions: Search and retrieval functionality is the basis functionality beside the data presentation and navigation in information services. We distinguish a variety of search functions:

Search and retrieval based on meta-properties: Components of media objects are of different importance for queries and especially for search. Some of them are crucial. Some of them are auxiliary. Distinguished components are considered to be meta-properties.

Retrieval on main properties: Retrieval is often based on main properties or associations represented in media objects. The meta-properties can be used for retrieval. Association-based retrieval follows associated types. Fuzzy retrieval allows the visitor to ask a question that is transferred or translated to a correct question. Retrieval can be also based on similarity functions such as functions for phonetic matching like soundex functions. Retrieval can also be based on special functions.

Search function can be combined with orientation functions.

Supporting functions: Visitors and actors working with or for the information service require a large number of functions which are typically the same as for most software systems.

Context-sensitive help: The visitor often needs help beside the general purpose help or the FAQ lists. Session objects allow to trace the reason for the help request and can support the user.

Privacy functions: Visitors often do not like to be tracked and monitored. For this reason, a clear statement which privacy policy is applied and privacy functions must be provided.

Tools: Information services development, evolution and maintenance is a complex task that must be supported by a variety of tools for content management, for link control, for maintenance of the site, for adaptation to new technology, for backups, security, mirroring, versioning, for site observation and diagnostics, for site extension and restructuring, for integration of foreign content and functionality, for mediators, and for wrappers.

Tours and trailers: Novel or casual visitors want to know what is the service providing, what facilities they can use, how they can navigate through the service, etc. Trailers and guided tours provide an impression on the service.

Other support functions allow the visitor to organise his or her workplace or workspace, to mark and to label data, to track their history of site usage, to introduce their own shortcuts, and to store their input information for later use.

Quality Assurance

Information services intend to meet the needs and expectations of users searching for certain information and services. Most German towns currently have their homepage, and most of them are static. Quality, and especially actuality, cannot be maintained in a simple fashion. We often observe that content for tourists is completely outdated or is misleading. In some WISs the same information demand is answered in a contradicting way in different pages of the same WIS. Therefore, quality management and especially actuality management is a crucial success factor of information services.

Missing content and missing functionality is one of the main pitfalls observed for information services. Such pages rarely meet all users' needs. For instance, university sites do not properly support either pupils or information brokers. The representation of such pages is oriented to internal needs. External users are not able to capture the content of the pages. The dialogues used for the sites are hard to comprehend. The user intending to look at or scan a page and the user exploring sites are not properly supported. Such users require an intuitive understanding of the 'mission' or the corporate identity of sites. The support for comprehension of the mental model of the page or the site can be based on metaphorical structures.

Information services are bound by the following quality criteria:

- Classical quality characteristics are completeness, correctness, actuality, right condensate, right presentation, aesthetics, originality, navigation, help, and search.
- Quality in use characteristics are effectiveness (task effectiveness, task completion, error frequency), productivity (task time, task efficiency, economic productivity, productive proportion, relative user efficiency), safety (user health and safety, safety of people affected by use of the system, economic damage, software damage), and satisfaction (satisfaction scale, satisfaction questionnaire and response, discretionary usage).
- Information logistics quality requires right information, to the right customer, in time, to the right place, in the right format, in the right tone and context.

Visitors must be able to judge which quality criterion is preserved. For instance, if visitors know about problems of actuality then they can use the content with care. If users know that service is provided by compromising privacy then they can change their behaviour whenever they like. Hidden click-stream analysis and broken privacy is a phenomenon observed nowadays for many information services.

Content presentation often uses metaphors such as

- the (cultural) journal for serious and selected information,
- the program journal for surveyed schedules,
- advertisement for selection on choice of provider,
- the supermarket basket for selection of data by the visitor,
- the phone book or other directories for monolithic ordering,
- the topic landscape for highlights, and
- the content landscape for annotation, preview, and quality survey.

6.7 Bibliographical Remarks

Our main source for website categorisation is the book [236]. An alternative categorisation has been developed in [664]. Our research has been condensed in the reprints [827, 844].

E-business and e-commerce systems have got a very wide attention in the literature. We thus reference only such sources that are directly concerned with our approaches, e.g., [255, 256, 554, 625, 689, 779]. The loan example is similar to [75, 398, 713]. The development of e-business websites that enhance existing business can be based on the the existing business models [351, 712, 709]. The development of novel e-business WIS is however a challenging task. The e-business bubble and failures and pitfalls of e-business applications have been widely discussed in newspapers. As far as we know, an analysis of failures is still missing (a good source is however the “Journal of Evolutionary Economics” [135]). It seems that novel business models have to start with novel approaches to business stories and thus storyboards and resulting models for

context and intention. [864] classified already in 1998 10 generic types of e-business models: e-shop, e-procurement, e-auction, third party marketplace, e-mail, virtual communities, value chain integrator, information broker, value chain service provider, and collaboration platforms. Later, [662] discussed 41 types of such business websites (see also [378, 633]).

Community websites have already been extensively studied before the advent of Web 2.0, e.g., [944]. Web 2.0 websites are user driven and content centered (see, for instance, GoogleAdSense, Flickr, Wikipedia, del.icio.us, musicstrands, root.net, 43thing, and blogs). They use optimised search engines, also provide content on the basis of pay-per-click and other strategies, are often combined into web services, and are based on paradigms such as participate-instead-of-be-attracted. Many techniques have already been incorporated, e.g., tagging, syndication, common usage of bookmarks, clicks, communities, tracking, data ownership, portability, etc. Web 2.0 is based on the LOW principle: *let others work* (in German AAL: andere arbeiten lassen). One lesson of community website development is its central orientation on the storyboard and users within their potential intentions.

Edutainment or e-learning systems have been widely discussed in the literature, e.g., [44, 372, 394, 709, 742]. Systematic construction of learning scenarios is the topic learning design [440]. They challenge developers by specific requirements for adaptation, context integration, learner typing, content proliferation, interfacing, etc. [615]. The DaMiT system [177, 428, 94, 795, 809] aims at developing technology components for advanced learning, i.e., *learning by doing, in collaborative groups, on demand, on own profile and portfolio*. Content object are extended open learning objects [516, 519]. Currently, e-learning is still based on blended learning didactics. However, the potential of edutainment lies in other didactics such as interactive or cooperative forms of learning [43, 796, 771, 291, 558].

Infotainment or information WIS are currently the majority of websites. They mainly follow conceptions for information presentation, e.g., [610]. Successful implementations have been developed on museum paradigms. Region and city information systems have been following this approach (e.g., [468]). A specific form of infotainment WISs are e-government portals (e.g., [45, 113, 226]). They include partially community WISs. They started with e-business elements as well. Currently e-business is of minor interest.

Game or entertainment websites are seldom data-intensive. Their functionality is typically very advanced. The storyboard approach is however also intensively used in an adapted form, e.g., [44, 254, 257, 270].

The chapter extends Chapter 2. As already mentioned, branding is an old marketing concept [445, 562, 341]. A brand can be extended to an informative model [848]. The classical WIS branding has been based on the association between provider and user (e.g., B(usiness), C(ustomer), A(dminstration)) and uses acronyms such as B2B (business to business). Our extension uses the provider-good-user-activity pattern for branding. It incorporates a spec-

ification of user groups by actors [747], a specification of activities via word fields or verb fields, and a specification of the goods.

Websites constantly evolve. This observation is not only valid for the content but also for functionality, presentation and the storyboard. Additionally, context and intention change quite frequently. Software aging and evolution [95, 120, 156, 319, 679, 638] counted changes in years. Information systems change more often [426]. WISs, however, have changes within quarters or months. Therefore, evolution and migration must be planned in advance. Moreover, the classical approaches such as big bang, chicken little, or butterfly are not entirely appropriate due to the holistic change for all six dimensions.

The best approach to change, evolution, migration, and in general modernisation is thus planning in advance. We may use the dependence of dimensions for each specific category [620] and order the dimensions into stages for the corresponding modernisation, e.g.,

- for e-business websites: (I) intention; (II) storyboard and context; (III) either functionality or content first and next the other; (IV) presentation;
- for edutainment: (I) intention; (II) content;
- for identity and community websites: (I) intention, storyboard and context coherently; (II) presentation; (III) either functionality or content first and next the other;
- for infotainment: (I) intention; (II) context; (II) storyboard; (IV) content and functionality; (V) presentation.

Key Messages

See **strategic analysis** and **usage analysis** for different classes of web information systems:

- e-business and e-commerce;
- community and group WIS;
- entertainment and gaming systems;
- identity and personal presentation WIS;
- learning and edutainment systems;
- information services.

Part III

Conceptual WIS Design – Rigorous Modelling of Web Information Systems and Their Layout with Web Interaction Types and Screenography



Web Interaction Types

The goal of this chapter is to introduce *Web Interaction Types*¹ as the major concept for conceptually modelling content and functionality of a Web Information System in an integrated way that seamlessly arises from the storyboard. As a means for conceptual modelling web interaction types will abstract from all details of implementation and also presentation addressing only data that is processed and the operations, by which these data are manipulated. That is, web interaction types link to the information production and consumption in the storyboard and the actions associated with the scenes.

From storyboarding we already obtain a hierarchy of scenarios, which ultimately leads to elementary scenes that are not further specified. Such an elementary scene defines an abstract location plus actions that can be initiated in that location. Furthermore, actions are associated with a specification of the information they consume and the information they produce plus a specification of context. This has been elaborated in detail in Chapter 3.

Now, the idea of web interaction types is to provide a conceptual model that provides a concrete, detailed specification of the data content associated with such locations together with a specification of the operations realising the actions on that location. The data content should capture the informa-

¹ Formerly, the notion *media type* was used instead of *Web Interaction Type*. The rationale behind this terminology was that a common understanding of “media” is a means for mass communication. Thus, the use of the term “media” suggests that the WWW has become such a means. Furthermore, “type” refers to the classification and formal abstraction from content and functionality of such communication means. It also includes the common understanding that a type defines instances, i.e., a media type defines a set of media objects as its instances, and these media objects actually represent concrete instantiations of the abstractly defined content and functionality. On the other hand, the term “media type” has been used simultaneously in the context of web application systems to refer simply to multi-media content, i.e., images, video or audio. Though this is a much narrower understanding of “media”, it is nonetheless a source of permanent confusion, so in this monograph we abandon the notion “media type”.

tion consumption of all associated actions, while the data production should become part of the specification of the operations. Thus, the primary purpose of web interaction types is to provide means for interface abstraction in web information systems, i.e., they describe which data shall be presented to the user in the scene to be supported and where this data comes from. The former aspect should lead to a definition of data types, while the latter links to defining queries on some underlying database(s).² This also enables the integrated handling of data on a global level, i.e., the data in the database(s), and on a local level, i.e., in the views defined by the queries. In addition, we even claim that the static navigation structure that should be implemented in the form of links on pages implementing the web interaction objects can be conceptually captured by the two-level data content specification.

Furthermore, for the tight coupling of operations with data content refining elementary scenes and the actions associated with them the idea of common user access can be exploited. This was already in the development of dialogue types, which provide interface abstractions for dialogue-centric information systems. That is, the general idea is that in order to start an operation, a user selects data out of the data presented to him or her. This should lead to a specification of selection types associated with operations, and by means of these selection types the required data consumption can be formalised. Then a user selects an operation that processes the selected data, but may also require additional input from the user. The execution of the operation as such will then require the execution of one or more database transactions operating on the global data layer, and produce a result that includes the information production. Furthermore, we can bring in the continuation of the plot of the storyboard by allowing operations to create new (and not only one) web interaction objects that enable the dynamic navigation to successor scenes.

Taking these ideas of views defined on underlying database(s) and operations on such views that induce database transactions and view initialisation together we capture the basic ingredients for interface abstraction, which is the primary focus of web interaction types. However, context injection is not yet handled by this. Therefore, we first approach an intermediate definition of *interaction types* in Section 7.1 capturing these ideas. We will also elaborate the instantiation of such interaction types by interaction objects. These show a close resemblance to dialogue objects with the difference that interaction objects now will contain static links and that we try to develop the model in a way that avoids close coupling with a particular underlying data model, as conceptual data modelling is not the focus of this monograph. Nonethe-

² We tacitly assume that in web information systems large amounts of data are processed, which justifies the assumption that databases are used for storing these data. Nonetheless, as we will elaborate in this chapter, the term “database” can be understood in a very broad sense capturing any kind of persistent storage of data.

less, we will illustrate a concretisation of the theory on the grounds of the Higher-Order Entity-Relationship Model.

This basic idea underlying web interaction types was adopted from the integration of database and dialogue objects in [716] and was present already in very early versions of the theory [222, 221, 720]. The approach to integrated querying was further developed in [707] leading to the consolidated version of web interaction types in [709, 727, 726], which builds the ground for a sophisticated architecture for the implementation of web interaction types [414].

Then the remaining sections of this chapter are dedicated to various extensions to the basic interaction types, which altogether define the complex concept of a web interaction type and its instances, the web interaction objects. The most important extensions concern *adaptivity* capturing restrictions arising from end-devices, communication channels or user preferences, and *granularity* capturing user preferences. These will be discussed in Sections 7.2 and 7.3, respectively. For adaptivity, the general idea is that restrictions on the size of objects to be communicated or presented are tackled by either decomposition or aggregation. In the former case an interaction object is replaced by a sequence of interaction objects with decreasing importance for the user. We will present two approaches dealing with such decompositions based on cohesion and proximity values, respectively. In the latter case aggregation functions are needed. For granularity the general idea is to provide various possibilities of hiding parts of the data with generic operations to navigate between coarser- and finer-grained presentations. Both adaptivity and granularity extensions exploit hierarchies of types. They further refine the picture of web interaction objects to families of such objects that are defined in two dimensions defined by adaptivity and granularity features.

Finally, Section 7.4 summarises the discussion in this chapter giving the final definition of web interaction types and elaborating the notion of web interaction schemata, i.e., sets of such types, and their semantics.

Adaptivity and granularity have been discussed in previous work such as [711, 726, 727], each time increasing the expressiveness of the concept of web interaction types.

7.1 Interaction Types

As announced above this section is dedicated to introduce conceptual means for interface abstraction, i.e., capturing information consumption and production on (elementary) scenes together with the actions and information production on them. That is, we will sequentially discuss abstract content modelling, coupling with databases, and the integrated specification of operations. The discussion will remain on an abstract, conceptual level avoiding any reference to web pages that can be used to implement these concepts, i.e., we completely abstract from such low-level interface considerations. In following

sections we will then present various extensions dealing with adaptivity to devices, channels, preferences and presentation.

7.1.1 Capturing Information Consumption

We now present the central concepts used for modelling the content and functionality of a WIS. For the content we assume that we have an underlying database. Abstracting from a specific data model the schema of this database can be described by *database types*. For instance, for a database schema defined in a variant of the Entity-Relationship model the database types would refer to the entity and relationship types, while in the case of an XML-based database, the database types would refer to trees. We will also distinguish *data types* as means for the conceptual description of domain values, though we will see that this distinction may be blurred.³ The available data types are defined by some (arbitrary) type system, which defines base types and type constructors. However, for the purpose of capturing static links we request that one of the case types is *URL*, a type of abstract url.

Example 7.1. Let an underlying type system be defined as

$$t = b \mid (a_1 : t_1, \dots, a_n : t_n) \mid \{t\} \mid [t] \mid \langle t \rangle.$$

Here b represents an arbitrary collection of *base types*, e.g., *BOOL* for Boolean values **T** and **F**, **1l** for a single value **1**, *TEXT* for text, *PIC* for images, *MPIC* for video data, *CARD* and *INT* for numbers, *DATE* for dates, *URL* for URL-addresses, *MAIL* for e-mail-addresses, etc. The constructors (\cdot) , $\{\cdot\}$, $[\cdot]$ and $\langle \cdot \rangle$ are used for records (or tuples), finite sets, finite lists, and finite multisets.

The next step is to introduce *interaction types* based on extended views over the database schema. This will allow us to apply completely different design criteria for the database schema and the *interaction schema*. One major facility used in interaction types is the possibility to create a navigation structure via URLs and links. We present an algebraic approach to querying and view definition following [707]. This will be elaborated in detail in Section 7.1.2.

Interaction types describe the content and functionality at a particular scene in the story space. We have seen that the content is defined by a view on some underlying database, so the core of an interaction type will be defined by a view, which roughly speaking is itself defined by a stored query. Furthermore, scenes are parameterised, so we will refer to the individual members of the view as *interaction objects*.

³ As a side note we would like to emphasize that the semantics is defined in such a way that schemata can be easily mapped to an XML representation [2]. This transformation is, however, not the reason for the particular definition of semantics.

Functionality is defined by adding operations to the interaction types. These operations have to be understood as realisations of the actions used in the storyboard.

Definition 7.2. A *view* V on a database schema \mathcal{S} consists of a view schema \mathcal{S}_V and a defining query q_V , which transforms databases over \mathcal{S} into databases over \mathcal{S}_V .

A *structure expression* results from a data type, in which the place of a base type may be occupied by a pair $\ell : M'$ with a label ℓ and the name M' of an interaction type. Replacing all such occurrences $\ell : M'$ in a structure expression exp by the base type *URL* defines the *content data type* associated with exp .

An *interaction type* has a name M and consists of a structure expression $cont(M)$, a defining query q_M such that $(\{t_M\}, q_M)$ defines a view, and a set of operations. Here t_M is the content data type associated with $cont(M)$.

Finite closed sets \mathcal{C} of interaction types define *content schemata*. Then a database \mathcal{D} over the underlying database schema \mathcal{S} and the defining queries determine finite sets $\mathcal{D}(M)$ of pairs (u, v) with URLs u and values v of type t_M for each $M \in \mathcal{C}$. We use the notion *pre-site* for the extension of \mathcal{D} to \mathcal{C} . The pair (u, v) will be called an *interaction object* in the pre-site \mathcal{D} .

However, pre-sites only define the set of *possible* interaction objects defined by a concrete database \mathcal{D} . At a certain point in time, only some of these interaction objects may be in use, i.e., a user works with the presentation of the content. Furthermore, the same interaction object may be used at the same time by different users. Therefore, we have to consider a multiset of interaction objects from the pre-site \mathcal{C} , the multiset of *open interaction objects*. Each open interaction object is associated with a user in a particular session.

7.1.2 Coupling with Databases

In principle we could use any query language. However, for our purposes the query language used in the views must be powerful enough to create navigation links, i.e., we must create URLs in the result of a query. Thus, our first task is to introduce such a query language.

The defining query of an interaction type may be expressed in any suitable query language, e.g., query algebra, logic or an SQL-variant. We shall outline a general algebraic approach following [707]. There it has been shown that any query algebra can be defined by operations defined from the underlying type system plus one generalized join-operation. This extends to rational tree types, hence is suitable for our case, as we may expand the links defined via values of type *URL* to obtain rational trees. On the other hand, it can be shown that such a query language is equivalent to a query language which allows to create URLs in a non-deterministic way.

Generic Query Algebras

Roughly speaking, a rational tree in our case would correspond to a page of infinite size which results from replacing all links by copies of the referenced page. As the navigation structure may contain cycles, the resulting page would be infinite.

Consider a trivial type denoted $\mathbb{1}$ and a Boolean type $BOOL$. Values of these types are $\mathbf{1}$ and \mathbf{T}, \mathbf{F} , respectively. There is no operation on $\mathbb{1}$, but for $BOOL$ we may consider the operations $\wedge : BOOL \times BOOL \rightarrow BOOL$ (conjunction), $\neg : BOOL \rightarrow BOOL$ (negation) and $\Rightarrow : BOOL \times BOOL \rightarrow BOOL$ (Implication). Furthermore, we consider two constants $\text{true} : \mathbb{1} \rightarrow BOOL$ and $\text{false} : \mathbb{1} \rightarrow BOOL$.

For tuple types we consider *projection* $\pi_i : t_1 \times \cdots \times t_n \rightarrow t_i$ and *product* $o_1 \times \cdots \times o_n : t \rightarrow t_1 \times \cdots \times t_n$ for given operations $o_i : t \rightarrow t_i$.

For set types we may consider \cup (union), $-$ (difference), the constant $\text{empty} : \mathbb{1} \rightarrow \{t\}$ and the *singleton* operation $\text{single} : t \rightarrow \{t\}$ with well known semantics. In addition, we consider structural recursion, which will be discussed in detail in the next subsection. We dispense with a discussion of the similar situation for list and multiset types.

For function types we consider *composition* $\circ : (t_2 \rightarrow t_3) \times (t_1 \rightarrow t_2) \rightarrow (t_1 \rightarrow t_3)$, *evaluation* $\text{ev} : (t_1 \rightarrow t_2) \times t_1 \rightarrow t_2$, and *abstraction* $\text{abstr} : (t_1 \times t_2 \rightarrow t_3) \rightarrow (t_1 \rightarrow (t_2 \rightarrow t_3))$. All these operations are standard.

For completeness assume an equality predicate $=_t : t \times t \rightarrow BOOL$ for all types t except function types and a membership predicate $\in_t : t \times \{t\} \rightarrow BOOL$. We shall also use a unique “forget”-operation $\text{triv} : t \rightarrow \mathbb{1}$ for each type t . Combining all the operations for all types of the type system gives all operations induced from the type system.

Let us now take a closer look at a powerful class of operations defined by the method of structural recursion [825]. For set types there are three natural constructors: the constant empty , the singleton operation and the union operation. In order to define an operation on a set type, say $\text{op} : \{t\} \rightarrow t'$, it is therefore sufficient to define it on the empty set, on singleton sets and on unions.

Formally, we define $\text{op} = \text{src}[e, g, \sqcup]$ with a value e of type t' , a function $g : t \rightarrow t'$ and a function $\sqcup : t' \times t' \rightarrow t'$. Then $\text{src}[e, g, \sqcup]$ is defined as follows:

$$\begin{aligned} \text{src}[e, g, \sqcup](\emptyset) &= e \\ \text{src}[e, g, \sqcup](\{x\}) &= g(x) \quad \text{for each } x \text{ of type } t, \text{ and} \\ \text{src}[e, g, \sqcup](X \cup Y) &= \\ &\quad \text{src}[e, g, \sqcup](X) \sqcup \text{src}[e, g, \sqcup](Y) \quad \text{for disjoint } X, Y \text{ of type } \{t\}. \end{aligned}$$

Analogously, for lists we have the empty list $[]$, a singleton operation giving $[x]$ and list concatenation $X.Y$. In this case we obtain the analogous definition

$$\begin{aligned}\mathbf{src}[e, g, \sqcup](\emptyset) &= e \\ \mathbf{src}[e, g, \sqcup]([x]) &= g(x) \quad \text{for each } x \text{ of type } t, \text{ and} \\ \mathbf{src}[e, g, \sqcup](X.Y) &= \mathbf{src}[e, g, \sqcup](X) \sqcup \mathbf{src}[e, g, \sqcup](Y) \text{ for each } X, Y \text{ of type } [t].\end{aligned}$$

Let us illustrate structural recursion by some more or less standard examples. First consider a function $f : t \rightarrow t'$ for arbitrary types t and t' . We want to “raise” f to a function $\mathbf{map}(f) : \{t\} \rightarrow \{t'\}$ by applying f to each element of a set. Obviously, we have

$$\mathbf{map}(f) = \mathbf{src}[\emptyset, \mathbf{single} \circ f, \sqcup].$$

Next consider a function $\varphi : t \rightarrow \text{BOOL}$. We want to define an operation $\mathbf{filter}(\varphi) : \{t\} \rightarrow \{t\}$, which associates with a given set the subset of all elements “satisfying the predicate” φ , i.e., elements that are mapped to \mathbf{T} . Then we may write

$$\mathbf{filter}(\varphi) = \mathbf{src}[\emptyset, \mathbf{if_then_else} \circ (\varphi \times \mathbf{single} \times (\mathbf{empty} \circ \mathbf{triv})), \sqcup]$$

with the function $\mathbf{if_then_else} : \text{BOOL} \times t \times t \rightarrow t$ with $(\mathbf{T}, x, y) \mapsto x$ and $(\mathbf{F}, x, y) \mapsto y$.

As a third example assume that t is a “number type”, on which addition $+ : t \times t \rightarrow t$ is defined. Then $\mathbf{src}[0, \mathbf{id}, +]$ with the identity function \mathbf{id} on t defines the sum of the elements in a set. In this way all the known aggregate functions of SQL (and more) can be defined by structural recursion.

In [707] it has been shown that the operations defined so far are sufficient to express operations such as `nest` and `unnest`.

Generalised Join Operators

In order to obtain also a generalised join it is a natural idea to exploit subtyping on the type system. This is a preorder \leq on the types.

Suppose, our collection of base types contains at least the type $\mathbf{1}$. BOOL may be identified with $\{\mathbf{1}\}$. Then subtyping can be defined in the standard way as the smallest preorder such that the following holds:

- For any type t we have $t \leq \mathbf{1}$.
- For set types (or list types, respectively) we have $\{t\} \leq \{t'\}$ (or $[t] \leq [t']$, respectively) iff $t \leq t'$ holds.
- For tuple types we have $t_1 \times \cdots \times t_m \leq t'_1 \times \cdots \times t'_n$ iff $t_{\sigma(i)} \leq t'_i$ holds for some injective $\sigma : \{1, \dots, n\} \rightarrow \{1, \dots, m\}$.

Then each subtype relation $t \leq t'$ defines an associated subtype function $\pi_{t'} : t \rightarrow t'$. Note that the projections in relational algebra are just such subtype functions. Indeed, t is the least common supertype of t_1 and t_2 ; $t_1 \bowtie_t t_2$ is a common subtype.

The following theorem is central for the definition of the general join [707].

Theorem 7.3. Consider a type system with the trivial type $\mathbb{1}$ as one of its base types and with constructors among the tuple, set and list constructors. If t is a common supertype of t_1 and t_2 with associated subtype functions $\pi_t^i : t_i \rightarrow t$, then there exists a common subtype $t_1 \bowtie_t t_2$ together with subtype functions $\pi_{t_1} : t_1 \bowtie_t t_2 \rightarrow t_1$ such that $\pi_t^1 \circ \pi_{t_1} = \pi_t^2 \circ \pi_{t_2}$ holds. Furthermore, for any other common subtype t' with subtype functions $\pi'_{t_i} : t' \rightarrow t_i$ with $\pi_t^1 \circ \pi'_{t_1} = \pi_t^2 \circ \pi'_{t_2}$ there is a unique subtype function $\pi : t' \rightarrow t_1 \bowtie_t t_2$ with $\pi_{t_i} \circ \pi = \pi'_{t_i}$.

For $t = \mathbb{1}$ we obtain $t_1 \bowtie_t t_2$ simply as the product $t_1 \times t_2$. With the existence of the *join types* $t_1 \bowtie_t t_2$ the join over t can be defined as in the relational case. For this let C_1 and C_2 be classes. We define

$$C_1 \bowtie_t C_2 = \{z : T_{C_1} \bowtie_t T_{C_2} \mid \exists z_1 \in C_1. \exists z_2 \in C_2. \pi_{t_1}(z) = z_1 \wedge \pi_{t_2}(z) = z_2\}.$$

Example 7.4. Consider $t_1 = \{b_1 \times \{b_2 \times b_3\} \times b_4\}$ and $t_2 = \{b_1 \times \{b_5 \times b_3\} \times b_6\}$ with the common supertype $t = \{b_1 \times \{b_3\}\}$. Then we obtain the join type

$$t_1 \bowtie_t t_2 = \{b_1 \times \{b_2 \times b_5 \times b_3\} \times b_4 \times b_6\}.$$

Handling URLs

The structures allowed by the definition of databases in the previous subsection are all finite. In fact, values can be represented as finite trees. A slight generalization would be to allow infinite trees, but of course only such infinite trees that can be represented in a finite way. For this we introduce *labels* ℓ . We extend any given type system in such a way allowing types to be adorned with labels and labels themselves to be treated in the same way as base types. Thus, our type system extends to

$$t = b | \ell | t_1 \times \cdots \times t_n | \{t\} | [t] | \ell : t.$$

Furthermore, we have to restrict ourselves to *well-defined* types. For this we require that for each label ℓ occurring within a type t —in a place where we could have a base type instead—some decorated type $\ell : t'$ must occur in t , too.

Values of such types with labels can be written as an infinite tree. [Figure 7.1 a\)](#) shows such a tree. We call a tree *rational* iff the number of different subtrees is finite. Then only rational trees will be allowed as values of well-defined types with labels. For our example, this means to restrict to values of the form

$$(n_1, a_1, (n_2, a_2, (\dots, (n_k, a_k, (\dots)))),$$

such that $n_i = n_j$ and $a_i = a_j$ holds for some i and j . In addition, we would like to add a constraint and require $i = 1, j = 3$, but this constraint must be added explicitly; it is not captured by the type system. [Figure 7.1 b\)](#) illustrates such a rational tree.

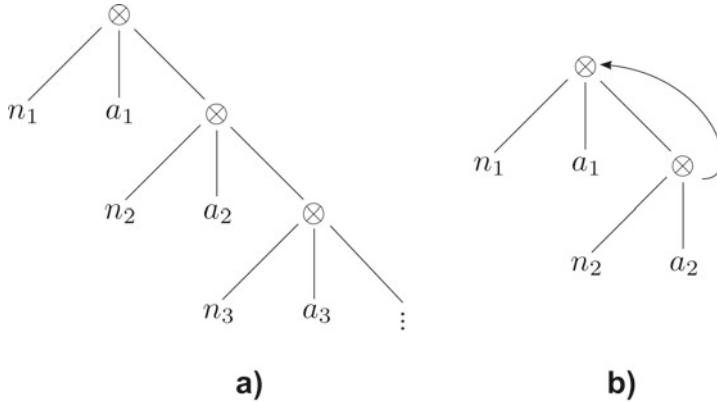


Fig. 7.1. Rational tree

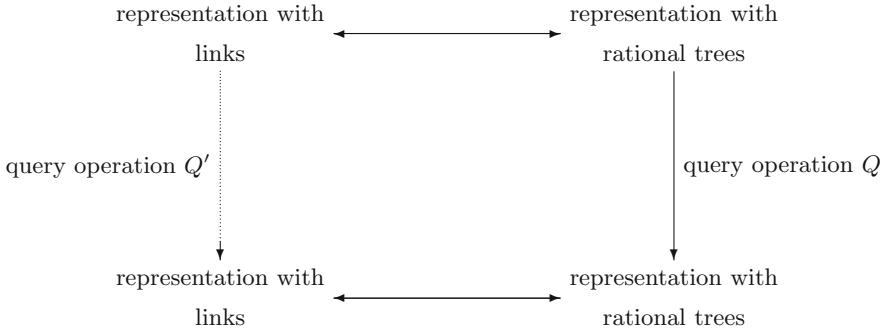
Since we restrict ourselves to well-defined types with labels, which can be written as rational trees, and allow only rational trees as values, we shall talk of *rational tree types*.

One important feature of rational tree types is that the query algebra outlined in the previous subsection extends naturally to rational tree types. Furthermore, as the representation with URLs can be regarded as a means to finitely represent rational trees, it can also be shown that we can replace the rational trees by the URLs, iff the query language is extended in such a way that it can create URLs and links. This can be done as follows:

- The operation `create_urls` transforms a set $\{v_1, \dots, v_m\}$ of values into a set $\{(u_1, v_1), \dots, (u_m, v_m)\}$ of pairs with new created URLs u_i of type *URL*;
- The operation `create_urls` also transforms a list $[v_1, \dots, v_m]$ of values into a list $[(u_1, v_1), \dots, (u_m, v_m)]$ of pairs with new created URLs u_i of type *URL*;
- The operation `create_url` transforms a value v of any type into a pair (u, v) with a new URL u .

Theorem 7.5. *Let S be a database schema, where the types of classes are rational tree types and let S' be an equivalent schema that uses the type *URL*, but no rational tree types. Then the result of an algebra query on S' with *URL* and link creation is the *URL-based representation* of the result of the same query applied to S and vice versa.*

Figure 7.2 illustrates the relationship between querying with URL creation and querying with rational trees.

**Fig. 7.2.** Handling URLs in queries

7.1.3 Entity-Relationship-Based Interaction Types

In principle, it is not important what kind of database we have as long as there exists a sufficiently powerful query mechanism that permits to define views. Nonetheless, let us now focus on a conceptual description of such databases using a data model close to the higher-order Entity-Relationship model (HERM) [825].

Higher-Order Entity-Relationship model

HERM is an extension of the well-known Entity-Relationship model. The major extensions we use here are:

- the use of nested attributes in order to enable already complex values in the database;
- the use of higher-order relationship types, i.e., relationship types over relationship types, which allow us to model complex aggregations;
- the use of clusters, i.e., disjoint unions of types, in order to model alternatives.

Let an underlying type system be defined as

$$t = b \mid (a_1 : t_1, \dots, a_n : t_n) \mid \{t\} \mid [t]$$

with the same conventions as in Example 7.1.

Definition 7.6. A *database type of level k* has a name E and consists of a set $comp(E) = \{r_1 : E_1, \dots, r_n : E_n\}$ of components with pairwise different role names r_i and database types (or clusters) E_i on levels lower than k with at least one database type of level exactly $k - 1$, a set $attr(E) = \{a_1, \dots, a_m\}$ of attributes, each associated with a data type $dom(a_i)$ as its domain, and a key $id(E) \subseteq comp(E) \cup attr(E)$. We shall write $E = (comp(E), attr(E), id(E))$.

A *cluster* of level k has a name E and consists of a set $\text{frag}(E) = \{f_1 : E_1, \dots, f_n : E_n\}$ of fragments with pairwise different fragment names r_i and database types (or clusters) E_i on levels at most k with at least one of the E_i of level exactly k .

A *database schema* is a finite set \mathcal{S} of database types and clusters such that for all $E \in \mathcal{S}$ and all $(r_i : E_i) \in \text{comp}(E)$ or $(f_i : E_i) \in \text{frag}(E)$, respectively, we also have $E_i \in \mathcal{S}$.

Following [825] the definition of *databases* over a given database schema \mathcal{S} is straightforward. However, here we employ partial mappings instead of total mappings in order to define tuples.

Definition 7.7. A *tuple* over a database type $E = (\text{comp}(E), \text{attr}(E), \text{id}(E))$ is a partial mapping t defined on $\text{comp}(E) \cup \text{attr}(E)$ such that the following conditions hold:

- $t(r_i : E_i)$ is either undefined or a tuple over E_i (for $(r_i : E_i) \in comp(E)$);
 - $t(a_j)$ is either undefined or a value in $dom(a_j)$ (for $a_j \in attr(E)$);
 - t is always defined on all elements of the key $id(E)$.

A *tuple* over a cluster E is a pair $(f_i : t_i)$ for some $(f_i : E_i) \in frag(E)$ and a tuple t_i over E_i .

A database \mathcal{D} over a database schema \mathcal{S} is an \mathcal{S} -indexed family $\{\mathcal{D}(E)\}_{E \in \mathcal{S}}$ of finite sets $\mathcal{D}(E)$ of tuples over E such that the following conditions hold:

- If E is a type, then for each $(r_i : E_i) \in \text{comp}(E)$ and each $t \in \mathcal{D}(E)$ for which $t(r_i : E_i)$ is defined $t(r_i : E_i) \in \mathcal{D}(E_i)$ holds;
 - If E is a cluster, then for each $(f_i : E_i) \in \text{frag}(E)$ and each $(f_i : t_i) \in \mathcal{D}(E)$ we have $t_i \in \mathcal{D}(E_i)$.

Furthermore, we can easily define a graphical representation for a database schema.

Example 7.8. We continue the on-line loan application from Example 3.9. For this application we obtain the following database type definitions:

`LOAN_TYPE = (\emptyset , { type, conditions, interest }, { type })`

CUSTOMER = (\emptyset , { customer_no, name, address, date_of_birth }, { customer_no })

```
HOME_LOAN = ({ type : LOAN_TYPE }, { loan_no, amount,
    interest_rate, begin, end, terms_of_payment }, { loan_no })
```

MORTGAGE = ({ type : LOAN_TYPE }, { mortgage_no, amount, disagio, interest_rate, begin, end, object }, { mortgage_no })

LOAN ≡ HOME-LOAN ⊕ MOBTGAGE

ACCOUNT = ($\{ \ell : \text{LOAN} \}$, $\{ \text{account_no}, \text{amount} \}$, $\{ \text{account_no} \}$)

ACCOUNT_RECORD = ({ a : ACCOUNT }, { record_no, type, amount, date }, { a : ACCOUNT, record_no })

$$\text{OWES} = (\{\text{who : CUSTOMER}, \text{what : LOAN}\}, \{\text{begin, end}\}, \{\text{who : CUSTOMER}, \text{what : LOAN, begin}\})$$

$$\text{SECURITY} = (\{\text{whose : CUSTOMER, for : MORTGAGE}\}, \{\text{value, object, type}\}, \{\text{whose : CUSTOMER, for : MORTGAGE, object}\})$$

$$\text{INCOME} = (\{\text{who : CUSTOMER}\}, \{\text{type, amount, frequency, account}\}, \{\text{who : CUSTOMER, account}\})$$

$$\text{OBLIGATION} = (\{\text{who : CUSTOMER}\}, \{\text{type, amount, frequency, account}\}, \{\text{who : CUSTOMER, account}\})$$

[Figure 7.3](#) provides a graphical representation of a database for this application. According to the common convention in Entity-Relationship diagrams we represented types on level 0 by rectangles and types on higher levels by diamonds. We use directed edges from a relationship type to each of its components, and from clusters to their components. Note that in this context a unary relationship type simply expresses that the number of components is one. In some cases – indicated by the presence of particular cardinality constraints – this expresses an IsA-relationship (see [825] for a detailed discussion of unary relationship types). Attributes and role names have been omitted in the schema. Clusters are represented by the \oplus symbol, but we omitted fragment names.

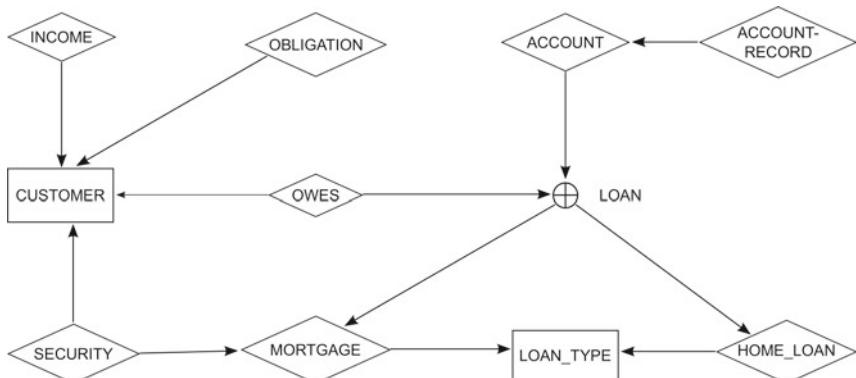


Fig. 7.3. Graphical representation of a database schema

To complete the structure definition of the underlying database we have to add static integrity constraints to the definition of a database schema. We refer to [825] for a detailed description of such constraints.

Example 7.9. Consider the scene home.loan in Example 3.15. For this scene the information consumption is the description of a particular loan type. So we get

$$\text{cont(home_loan)} = (\text{type : STRING, conditions : STRING, } \\ \text{interest : \{} (\text{amount : CARD, rate : FLOAT}) \}) .$$

In this case the defining query is $q_{\text{home_loan}} = \text{create_urls(LOAN_TYPE)}$.

Alternatives to Query Algebras

For each type $R \in \mathcal{S}$ we have a representing type t_R . In an \mathcal{S} -database db each type $R \in \mathcal{S}$ gives rise to a finite set $db(R)$ consisting of pairs (i, v) with i of type ID and v of type t_R . Using this we may set up a powerful query language adapting a logic programming approach as in [3].

Thus, a query will be expressed as a set of rules (precisely: a sequence of such sets). Evaluating the rule body on a given database will result in bindings of variables, and the corresponding bindings for the head together with the creation of new identifiers for unbound variables results in an extension to the database. These extensions are iterated until a fixed point is reached.

In order to formalise this, assume to be given countable sets of variables V_t for each type t . These sets are to be pairwise disjoint. Variables and constants of type t are *terms* of that type. In addition, each $R \in \mathcal{S}$ is a term of type $\{(ident : ID, value : t_R)\}$, and for each variable i of type ID (and URL , respectively) there is a term \hat{i} of some type $t(i)$. If τ_1, \dots, τ_k are terms of type t , then $\{\tau_1, \dots, \tau_k\}$ is a term of type $\{t\}$, and if τ_1, \dots, τ_k are terms of type t_1, \dots, t_k , respectively, then $(a_1 : \tau_1, \dots, a_k : \tau_k)$ is a term of type $(a_1 : t_1, \dots, a_k : t_k)$.

If τ_1, τ_2 are terms of type $\{t\}$ and t , respectively, then $\tau_1(\tau_2)$ is a positive literal (also called a *fact*) and $\neg\tau_1(\tau_2)$ is a negative literal. If τ_1, τ_2 are terms of the same type t , then $\tau_1 = \tau_2$ is a positive literal and $\tau_1 \neq \tau_2$ is a negative literal. A *ground fact* is a fact without variables.

A *rule* is an expression of the form $L_0 \leftarrow L_1, \dots, L_k$ with a fact L_0 (called the *head* of the rule) and literals L_1, \dots, L_k (called the *body* of the rule), such that each variable in L_0 not appearing in the rule's body is of type ID or URL , respectively. A *logic program* is a sequence $P_1; \dots; P_\ell$, in which each P_i is a set of rules.

Finally, a *query* Q on \mathcal{S} is defined by a type t_Q and a logic program \mathcal{P}_Q such that a variable ans of type $\{(url : URL, value : t_Q)\}$ is used in \mathcal{P}_Q . A *boolean query* can be described as a query Q with type $t_Q = \mathbb{1l}$. Alternatively, as we are not interested in creating a URL for the answer, we can simplify the approach above and consider a logic program, in which a variable ans of type $BOOL = \{\mathbb{1l}\}$ appears.

Example 7.10. Consider a query Q with type t_Q defined as

$$\begin{aligned} (\text{type} : STRING, \text{conditions} : STRING, \text{interest} : STRING, \\ \text{amount} : CARD, \text{disagio} : RATIONAL, \text{interest_rate} : RATIONAL, \\ \text{object} : STRING, \text{securities} : \{ (\text{value} : RATIONAL, \\ \text{object} : STRING, \text{type} : STRING) \}, \text{customers} : \{ (\text{income} : \\ \{ (\text{type} : STRING, \text{amount} : CARD, \text{frequency} : STRING) \} , \\ \text{obligations} : \{ (\text{type} : STRING, \text{amount} : CARD, \\ \text{frequency} : STRING) \} \}) \}) \end{aligned}$$

Assume the database schema \mathcal{S} contains at least the types LOAN_TYPE, CUSTOMER, MORTGAGE, OWES_MORTGAGE, SECURITY, INCOME, and OBLIGATION with the following representing types:

$$\begin{aligned} t_{\text{LOAN_TYPE}} &= (\text{type} : STRING, \text{conditions} : STRING, \text{interest} : STRING) \\ t_{\text{CUSTOMER}} &= (\text{customer_no} : CARD, \text{name} : STRING, \text{address} : STRING, \\ &\quad \text{date_of_birth} : DATE) \\ t_{\text{MORTGAGE}} &= (\text{type} : ID, \text{mortgage_no} : CARD, \text{amount} : CARD, \\ &\quad \text{disagio} : RATIONAL, \text{interest_rate} : RATIONAL, \\ &\quad \text{begin} : DATE, \text{end} : DATE, \text{object} : STRING) \\ t_{\text{OWES_MORTGAGE}} &= (\text{who} : ID, \text{what} : ID, \text{begin} : DATE, \text{end} : DATE) \\ t_{\text{SECURITY}} &= (\text{whose} : ID, \text{for} : ID, \text{value} : CARD, \text{object} : STRING, \\ &\quad \text{type} : STRING) \\ t_{\text{INCOME}} &= (\text{who} : ID, \text{type} : STRING, \text{amount} : RATIONAL, \\ &\quad \text{frequency} : RATIONAL, \text{account} : CARD) \\ t_{\text{OBLIGATION}} &= (\text{who} : ID, \text{type} : STRING, \text{amount} : RATIONAL, \\ &\quad \text{frequency} : RATIONAL, \text{account} : CARD) \end{aligned}$$

Then the following logic program \mathcal{P}_Q will produce an anonymised set of mortgages:

$$\begin{aligned} M_1(t, c, i, a, d, p, o, S, C, i_m) &\leftarrow \text{MORTGAGE}(i_m, (i_\ell, n, a, d, p, b, e, o)), \\ &\quad \text{LOAN_TYPE}(i_\ell, (t, c, i)); \\ \hat{S}(v, o, t') &\leftarrow \text{SECURITY}(i_s, (i_c, i_m, v, o, t')), \\ &\quad M_1(t, c, i, a, d, p, o, S, C, i_m). \\ \hat{C}(I, O, i_c) &\leftarrow \text{CUSTOMER}(i_c, (n', n, a, db)), \\ &\quad \text{OWES_MORTGAGE}(i_o, (i_c, i_m, b, e)), \\ &\quad M_1(t, c, i, a, d, p, o, S, C, i_m); \\ \hat{I}(t', a', f) &\leftarrow \hat{C}(I, O, i_c), \\ &\quad M_1(t, c, i, a, d, p, o, S, C, i_m), \\ &\quad \text{INCOME}(i_i, (i_c, t', a', f, ac)). \end{aligned}$$

$$\begin{aligned}
\hat{O}(t', a', f) &\leftarrow \hat{C}(I, O, i_c), \\
M_1(t, c, i, a, d, p, o, S, C, i_m), \\
\text{OBLIGATION}(i_o, (i_c, t', a', f, ac)); \\
M_2(t, c, i, a, d, p, o, \hat{S}, D) &\leftarrow M_1(t, c, i, a, d, p, o, S, C, i_m); \\
\hat{D}(\hat{I}, \hat{O}) &\leftarrow \hat{C}(I, O, i_c); \\
\text{ANS}(u, (t, c, i, a, d, p, o, \hat{S}, \hat{D})) &\leftarrow M_2(t, c, i, a, d, p, o, \hat{S}, D).
\end{aligned}$$

A program $P_1; \dots; P_\ell$ is evaluated sequentially. Each set of rules is evaluated by computing an *inflationary fixed point*. That is, start with the set of ground facts given by the \mathcal{S} -database db . Whenever variables in the body of a rule can be bound in a way that all resulting ground literals are satisfied, then the head fact is used to add a new ground fact. Whenever variables in the head cannot be bound in a way that they match an existing ground fact, the variables of type *ID* and *URL* will be bound to new identifiers or URLs, respectively.

7.1.4 Operations on Interaction Types

Conceptual abstraction of database behaviour is achieved via *operations* associated with database types. These operations can be described in a way known from programming languages.

Definition 7.11. An *operation* on a database type C consists of a *signature* and a *body*. The *signature* consists of an operation name O , a set of input-parameter/type pairs $i_i :: T_i$ and a set of output-parameter/type pairs $o_j :: T'_j$. The *body* is recursively built of the following constructs:

- *assignment* $x_E := exp$, where x is a variable representing the content of the type E itself or a local variable (including the output-parameters), and exp is an expression of the same type as x_E ,
- *local variable declaration* `LET x : t IN S`,
- *sequencing* $S_1 ; S_2$,
- *branching* `IF P THEN S1 ELSE S2 ENDIF`,
- *bounded parallelism* $S_1 \parallel S_2$,
- *unbounded parallelism* `FORALL x WITH φ DO S ENDDO;`
- *operation call* $E' :- O'(in : exp'_1, \dots, exp'_j, out : x'_1, \dots, x'_i)$, where O' is an operation on the type E' with compatible signature, and
- non-deterministic *selection* of values `NEW.f(x)`, where f is a selector on E .

Modelling Functionality

In order to model the required functionality we also add operations to interaction types. This is completely analogous to the d-operations on dialogue

types [716]. That is, operations on interaction types do not directly manipulate data, but use the operations on the underlying database for this purpose, the syntax of which is the same as for operation calls:

$$E' \text{ :- } O'(\text{in} : exp'_1, \dots, exp'_j, \text{out} : x'_1, \dots, x'_i),$$

where O' is an operation on the database type E' . All other constructs used in operations on database types except assignments can be used in the same way.

Another peculiarity of operations on interaction types that distinguish them from operations on database types is the fact that these operations are meant to be executed in the presence of an interaction object. According to our definition of pre-site the data content of an interaction object of type M is a value of type t_M . This value is the assumed to be presented to the user in one or another form. Furthermore, the data content type t_M comprises all information consumption for all operations associated with M . In particular, if an operation O on M is to be executed, most of the input required by O stems from the data content, i.e., from the presented value of type t_M , and only very little additional data has to be gathered from the user by means of input to O .

Therefore, we can precisely specify which part of the data of type t_M that is presented to the user must be selected in order to execute a particular operation. Such data selection can be considered to be realisable by a low-level function at the user interface. Selection from a type t_M can be defined recursively as follows:

- If t_M is a record type $(a_1 : t_1, \dots, a_n : t_n)$, then we may request that some of the fields indicated by the label names a_1, \dots, a_n must be selected, say a_{i_1}, \dots, a_{i_k} . This selection can be expressed by a type $(a_{i_1} : t'_{i_1}, \dots, a_{i_k} : t'_{i_k})$ with $\{i_1, \dots, i_k\} \subseteq \{1, \dots, n\}$. Here the t'_{i_j} are types describing which data must be selected from the value of type t_i presented to the user.
- If t_M is a list type $[t]$, then some of the values in the presented list may be selected, that is the selected value is of type $[t']$, where t' is a type describing which data must be selected from the (selected) value of type t presented to the user. If only a single value is to be selected, it will be of type t' .
- Analogously, if t_M is a set type $\{t\}$, then some of the values in the presented finite set⁴ may be selected, that is the selected value is of type $\{t'\}$, where t' is a type describing which data must be selected from the (selected) value of type t presented to the user. If only a single value is to be selected, it will be of type t' .

⁴ Actually, it can be assumed that the presentation is in the form of a list without repetitions, and order in that list is irrelevant.

- Also, if t_M is a multiset type $\langle t \rangle$, then some of the values in the presented finite multiset⁵ may be selected, that is the selected value is of type $\langle t' \rangle$, where t' is a type describing which data must be selected from the (selected) value of type t presented to the user. If only a single value is to be selected, it will be of type t' .
- If t_M is a base type, then the value may be selected or not, i.e., the selected value is of type t_M or of type $\mathbb{1}$. This also covers the case of references that are represented by values of type URL .
- Finally, if the type system provides a constructor for disjoint unions $(a_1 : t_1) \oplus \dots \oplus (a_n : t_n)$, this can be used to model alternative choices in the selection, i.e., the selection type can be a union type $(a_1 : t_1) \oplus \dots \oplus (a_n : t_n)$, in which each of the types t_i is built according to the possibilities discussed before.

Summarising this discussion on value selection, we can define a *selection type* $sel(O)$ of an operation O associated with an interaction type M as a supertype of the data content type t_M . In order to provide sufficient flexibility it is advisable that the underlying type system provides a trivial type $\mathbb{1}$ as well as a union type constructor \oplus . This gives rise to a selection function $sel : dom(t_M) \rightarrow dom(sel(O))$. However, it should be noted that in case of lists, sets and multisets this function maps a finite list, set or multiset to a sublist, subset or submultiset, respectively. Furthermore, which elements from a list, set or multiset are selected depends on the user, i.e., the function sel is not a priori defined, but can differ in the application.

Other input requested from the user can be specified by input signature in the same way as for operations on the database. How this additional input is gathered from the user is left to the implementation. It is well possible that common techniques such as *dialogue boxes* can be exploited for this purpose.

Furthermore, operations on interaction types are meant to capture dynamic navigation as defined in the storyboard. For this, the following two operations are permitted:

- **CLOSE** closes the interaction object, from which the operation has been started, and
- **OPEN $M(\varphi)$** , where M is the name of an interaction type and φ is a selection condition that selects a unique value of type t_M out of a set of such values.

As the defining query of an interaction type M defines a set of values of type t_M , then operation **OPEN $M(\varphi)$** creates a well-defined new open interaction object of type M .

In summary, an operation on an interaction type consists of an operation name signature, i.e., name, input-parameters and output-parameters and selection type, which is a supertype of the content data type defined by M , and

⁵ Actually, it can be assumed that the presentation is again in the form of a list with possible repetitions, but order in that list is irrelevant.

a body that is defined via operations accessing the underlying database and operations to open and close interaction objects.

Definition 7.12. An *operation* on an interaction type M consists of a *signature* and a *body*. The *signature* consists of an operation name O , a selection type $sel(O)$, which is a supertype of the type defined by $cont(M)$, a set of input-parameter/type pairs $\iota_i :: T_i$, a set of output-parameter/type pairs $\omega_j :: T'_j$. The *body* is recursively built of the following constructs:

- *operation call* $E' :- O'(\text{in} : exp'_1, \dots, exp'_j, \text{out} : x'_1, \dots, x'_i)$, where O' is an operation on the database type E' with compatible signature,
- *local variable declaration* $\text{LET } x : t \text{ IN } S$,
- *sequencing* $S_1 ; S_2$,
- *branching* $\text{IF } P \text{ THEN } S_1 \text{ ELSE } S_2 \text{ ENDIF}$,
- *bounded parallelism* $S_1 \parallel S_2$,
- *unbounded parallelism* $\text{FORALL } x \text{ WITH } \varphi \text{ DO } S \text{ ENDDO}$;
- non-deterministic *selection* of values $\text{NEW}.f(x)$, where f is a selector on E ,
- *CLOSE*, and
- $\text{OPEN } M(\varphi)$, where M is the name of an interaction type and φ is a selection condition that selects a unique value of type t_M out of a set of such values.

Example 7.13. Let us build upon Example 3.9 from Section 3.1.4 using the database schema in Example 7.8. Let us concentrate on the scene INCOME and the action enter_income_details.

Regarding the data content this could be specified as

$$cont(\text{INCOME}) = (\text{requested_loan} : t_{ql}, \text{customer_data} : t_{qc}, \text{income_data} : t_{qi})$$

with

$$t_{ql} = (\text{id} : ID, \text{amount} : CARD, \text{details} : LOAN_DETAILS),$$

where the first field id contains the identifier of the loan under preparation that is already stored in the database, the second field contains the requested amount. All further details are shifted to a different interaction type $LOAN_DETAILS$ that is reachable via the link details in the third field.

$$t_{qc} = (\text{customer_no} : CARD, \text{name} : STRING)$$

contains the customer information for the loan, and

$$\begin{aligned} t_{qi} = & (\text{reg_income} : FLOAT, \text{reg_expenditure} : FLOAT, \\ & \text{accounts} : \{(\text{account_no} : CARD, \text{institution} : STRING)\}). \end{aligned}$$

Queries can be defined as discussed before in this section. However, it should be noted that income data may not yet be available for the customer

in the database. In this case the values for regular income and regular expenditure will be set to 0 as default, and the accounts would be set to \emptyset . Also loan details may not yet all exist, in which case $\text{cont}(\text{LOAN_DETAILS})$ should provide the possibility to have optional fields, which can be obtained using the trivial type $\mathbb{1}$ and exploit disjoint unions.

A value of type t_{INCOME} corresponding to the particular customer applying for the loan and the data of the loan entered so far will be presented to the user. Then the details of the income can be changed by the user, i.e., the total regular income can be entered or updated, same for the total regular expenditure, and the accounts for evidence of this information can also be given. Usually, it can be assumed that some kind of form interface is available for capturing this low-level interface functionality, yet the model of interaction objects abstracts from such implementation details.

However, this enables us to define the selection type of the operation `enter_income_details` as

$$\text{sel}(\text{enter_income_details}) = (\text{income_data} : t_{qi}),$$

i.e., the operation will take the updated income details from the user as input. There is no need for additional input. Then the operation will simply call the database transaction `enter_income` to update the database with the selected information. Thus, the body of the operation `enter_income_details` is simply

`INCOME :- enter_income(in : s),`

where s is the selected value of type $\text{sel}(\text{enter_income_details})$.

Generic Functionality

According to the model of interaction types that we developed so far, the specification of interaction types based on the elementary scenes and actions in the storyboard is the major design activity on the conceptual layer – disregarding for the moment the various extensions to the concept that will be introduced and discussed in the following sections. This suggests that the operations associated with interaction types all result from refining actions in the storyboard.

However, this would be too narrow-minded, as the interface to a web information system would require additional standard operations. Such operations must be present in every scene, so we extend the model of interaction types by generic operations that are part of every interaction type without need to be explicitly specified. The following standard operations are of particular interest in web information systems:

- *Aggregation operations* are used for the generation of aggregated data. They are useful especially in the case of insufficient space or for the display of complementary, generalised information after terminating a task. In general, we understand aggregation as a lossy means to compact data

content. In terms of the query algebra we discussed early in this chapter this requires the usage of structural recursion. We have seen that selection (`filter`), element-wise mapping (`map`), but also summation, averaging, etc. can be expressed by structural recursion. Therefore, we define that an *aggregation operator* on type t is an algebra operator that involves at least one application of structural recursion.

Then an *aggregation operation* Agg on an interaction type M has selection type $\mathbb{1}$, i.e., nothing must be selected, and no additional input and output signature. Furthermore, the body of Agg is simply

OPEN $M_O(\text{val} = op(v))$,

where $op : t_M \rightarrow t_{M_O}$ is an aggregation operator on the content data type of M , v is the value of type t_M of the actual interaction object, val denotes the value of the content data type of M_O , and the interaction type M_O is defined in the same way as M , but with a content data type t_{M_O} that is given by means of the operator op .

We will also discuss automatic aggregation in connection with adaptivity in Section 7.2.

- *Generalisation operations* are used to switch to a more coarse-grained presentation of data content. The roll-up operation in [9] as well as slicing and grouping known from OLAP systems are special generalisation operations. These functions are connected to the presence of hierarchical versions, between which a user can switch. These hierarchies will be discussed in connection with granularity in Section 7.3.
- *Specialisation operations* are inverse to the generalisation operations. They are used to switch to a more fine-grained presentation of data content. The drill-down operations known from data warehouses provide a typical example. Same as generalisation operations they are connected to the presence of hierarchical versions, between which a user can switch, and thus will be discussed in connection with granularity in Section 7.3. In an implementation a specialisation operation may give rise to additional queries to the database to obtain more details for aggregated data.
- *Reordering operations* are used for the rearrangement of the data content. Examples are pivoting, dimension destroying, pull and push operations [9] and the rotate operation. We will discuss reordering in connection with the ordering extension of interaction types in Section 8.1.
- *Browsing operations* are connected to the presence of static navigation links between interaction objects. Following such links allows the user to explore additional information and execute additional operations on the linked interaction types. This enables a fine-tuned modelling of scenes distinguishing between global and local navigation. Furthermore, it may be used for detailed views within a scene.

In this case the name of the operation is simply *browse*, and the selection type $\text{sel}(\text{browse})$ is equivalent to a union type $(r_1; URL) \oplus \dots \oplus (r_k; URL)$, each of the components of which correspond to the reference links in the

interaction type. The body is defined as

$\text{LET } s = (r_i : j) \text{ IN OPEN } M_i(id = j),$

where s is the selected value of type $\text{sel}(\text{browse})$, M_i is the name of the interaction type in the link $r_i : M_i$ in $\text{cont}(M)$ and j is the identifier of an interaction object of type M_i .

- *Sequentialisation operations* are used for following links that result from splitting the data content of an interaction object. This may be the consequence of adaptation to channels, end-devices and user preferences as we will discuss in Section 7.2. Thus, sequentialisation is similar to browsing using the operation name *follow* and the selection type *URL* referring to the unique successor object. The body is defined as

$\text{OPEN } M'(id = s),$

where s is the selected value of type $\text{sel}(\text{follow})$, M' is the name of the successor interaction type and j is the identifier of an interaction object of type M' .

- *Join operations* provide an alternative to the browsing and sequentialisation operations, as they are used to construct more complex interaction objects by expanding references. Thus, as for browsing operations the selection type is equivalent to a union type $(r_1; \text{URL}) \oplus \dots \oplus (r_k; \text{URL})$, each of the components of which corresponds to the reference links in the interaction type, and the name of the operation is *join*. The body is

$\text{CLOSE} \parallel \text{LET } s = (r_i : j) \text{ IN OPEN } M(id = i) \bowtie_{r_i} M_i(id = j)$

where i is the url of the actually open interaction object, M_i is the name of the interaction type in the link $r_i : M_i$ in $\text{cont}(M)$, and j is the identifier of an interaction object of type M_i . The name of the resulting interaction type is implicitly defined by M and $r_i : M_i$.

- *Link operations* are useful whenever the user is required to imagine referring to the contextual information associated with the interaction object. In case of the navigation context this refers to details of the history of the interaction before entering the current scene that is represented by this interaction object. We will briefly discuss the contextual extension to interaction types in Section 8.1, but the major discussion referring to higher-level web interaction types will be postponed to Chapter 8.2.
- *Search operations* are used to enable the user to create additional information that may help in deciding how to proceed with the navigation through the web information system. If MS is the name of an interaction type representing the entry scene of a search system, then there is only a unique interaction object of type MS . Thus, the operation *search* has only a trivial selection type $\mathbb{1}$ and a body $\text{OPEN } M(\text{true})$. All further details depend on the specific search system.
- *Presentation operations* are used to change the presentation of the data. This is done by exploiting different measuring systems and presentation

options that will be discussed in more detail as extensions to interaction types in Section 8.1.

- *Visualisation operations* are similar to presentation operations with the specific purpose to support the visualisation of the data content of the interaction object. We will discuss visualisation in the context of screenography and content presentation in Chapters 9 and 10.

7.1.5 Alternative Form-Based Approaches

Let us now briefly examine some alternative approaches to interface abstraction in web information systems. We already mentioned form-based approaches, which we will look at in more detail. Furthermore, other web development methods such as WebML [144], UML-based methods [158], WSDM [184], HERA [333] and OOHDM [762] take a direct, conceptual approach to modelling interaction with a WIS, but surprisingly none refers explicitly to forms. In particular, there is no coupling with databases and no generation of links as part of querying. Furthermore, there are direct page-based interaction descriptions, e.g., in [923], the inadequacy of which we already discussed in Chapter 3.

A decade ago building form-based interfaces to Information Systems led to various proposals, the work in [189] being a good representative for this. The work in [196, 195] presents a specific abstract approach to form-based interfaces that is not linked to any particular WIS development method.

Let us briefly explore how the conceptual model of interaction types captures forms in an abstract way. Interaction types are defined as generalised views. That is, assuming some underlying database schema, a view is defined by a query on that schema. Evaluating it on a database will result in a set of objects, the interaction objects, each of which represents a piece of content presented to a user. The most important extension to this concept of view to turn it into an interaction type is the association of operations. Then a user can select a portion of the content presented to him, enter additional data, and start an operation, which will update the underlying database and create new interaction objects. This approach actually captures the gist of the object action principle in common-user access [339].

Abstraction from Low-Level Form Processing

Let us start with looking at the shape of forms. Usually, we find several fields, in which users can (or must) enter data. These fields can be accompanied by some text or other information, but this is merely needed to let a user know what the field is used for. In case the data to be entered must be taken from a finite list of options, selection menus can be provided, which have to be treated as just a representation choice for a field. Similarly, check-boxes and buttons that allow to make an exclusive or non-exclusive choice between

several options, are also nothing more than a more user-friendly representation of a particular input field.

In WISs forms are embedded in web pages, so there is even more surrounding information available. Therefore, a form can be represented by a complex value. A form can be empty or filled-in; both cases differ from each other just by the representing complex value. We can abstract from such complex values using a type system, e.g., (using abstract syntax):

$$\begin{aligned} t = & \mathbb{1} | b | (a_1 : t_1, \dots, a_n : t_n) | \{t\} | \langle t \rangle \\ & | [t] | (a_1 : t_1 (\oplus \dots \oplus (a_n : t_n)). \end{aligned}$$

Here b represents a not further specified set of base types, each associated with a countable domain, i.e., if INT is a base type, then $\text{dom}(\text{INT}) = \mathbb{Z}$ specifies that the associated domain is the set of integers. $\mathbb{1}$ is a further base type with a trivial domain, i.e., $\text{dom}(\mathbb{1}) = \{\perp\}$ is a singleton set. Furthermore, (\cdot) , $\{\cdot\}$, $\langle \cdot \rangle$, $[\cdot]$ and \oplus represent constructors for record, finite set, multiset (or bag), list, and disjoint union types. When using these constructors, we use label names a, a_i in them to distinguish the various components.

We may use these constructors to create other types.

Example 7.14. Take the familiar example of a registration form for a conference, in which we would have to fill in first name, last name, affiliation, postal address, phone, fax, email, plus a choice between regular and student participant, a choice between presenter (in which case the paper numbers have to be entered) and non-presenter, the number of additional proceedings volumes, and the number of additional banquet tickets. This leads to the following type for the registration data:

$$\begin{aligned} & (\text{first_name} : (\text{filled_in} : \text{STRING}) \oplus (\text{empty} : \mathbb{1})), \\ & (\text{last_name} : (\text{filled_in} : \text{STRING}) \oplus (\text{empty} : \mathbb{1})), \\ & (\text{affiliation} : (\text{filled_in} : \text{STRING}) \oplus (\text{empty} : \mathbb{1})), \\ & (\text{address} : (\text{filled_in} : \text{STRING}) \oplus (\text{empty} : \mathbb{1})), \\ & (\text{phone} : (\text{filled_in} : \text{PHONE}) \oplus (\text{empty} : \mathbb{1})), \\ & (\text{fax} : (\text{filled_in} : \text{PHONE}) \oplus (\text{empty} : \mathbb{1})), \\ & (\text{email} : (\text{filled_in} : \text{EMAIL}) \oplus (\text{empty} : \mathbb{1})), \\ & (\text{student?} : (\text{filled_in} : \{\mathbb{1}\}) \oplus (\text{empty} : \mathbb{1})), \\ & (\text{papers} : \{\text{INT}\}), \\ & (\text{proceedings?} : (\text{filled_in} : \text{NAT}) \oplus (\text{empty} : \mathbb{1})), \\ & (\text{tickets?} : (\text{filled_in} : \text{NAT}) \oplus (\text{empty} : \mathbb{1})) \end{aligned}$$

Here, STRING , PHONE , EMAIL and NAT are base types with the obvious meaning representing character strings, phone numbers, email addresses, and natural numbers. Each component of this record type corresponds to an

input field disregarding its presentation, and each field can be empty or filled in. In the case of the papers field the empty field corresponds to the value \emptyset . The form can be presented in various ways, e.g., the ‘student?’ entry could be represented by a checkbox, and the ‘papers’ entry by a list of entry fields.

Low level forms processing would check the correct typing of any input in case of submitting the data, which would be iterated until the input data is correctly typed. We can ignore this iteration and assume that when a filled-in form is submitted, we get a value of the specified type above.

In general, we could treat all data presented at a page as a complex value. Only the parts that can be changed could be represented as a form.

Example 7.15. The value of the type in Example 7.14

```
(first_name : (empty : ⊥), last_name : (empty : ⊥),
affiliation : (empty : ⊥), address : (empty : ⊥),
phone : (empty : ⊥), fax : (empty : ⊥),
email : (empty : ⊥), student? : (empty : ⊥), papers : ∅,
proceedings? : (empty : ⊥), tickets? : (empty : ⊥))
```

represents an empty form as it may be presented to a user, whereas the value

```
(first_name : (filled_in : “Bill”), last_name : (filled_in :
“Miller”), affiliation : (filled_in : “Clark University”),
address : (empty : ⊥), phone : (empty : ⊥),
fax : (empty : ⊥), email : (filled_in : bmiller@hatmail.com),
student? : (filled_in : ∅), papers : {35, 71},
proceedings? : (filled_in : 0), tickets? : (filled_in : 1))
```

represents a filled-in form of a regular participant presenting two papers and requesting one additional banquet ticket. No address, phone nor fax was entered.

So, the first step in abstracting from low-level forms processing within a WIS is to assume a user is presented with a complex value of some type, which he may change and submit. In particular, we can assume that the submitted value has the correct type.

With respect to types the form-based approach is similar to interaction types, though the coupling with databases and the generation of links by this way are unique to the concept of interaction types.

Operations

Furthermore, we may provide the possibility of choosing between several operations that can be performed with the submitted data. Again, how exactly

an operation is selected is of no importance. Finally, we may allow a user to highlight some of the input data, which corresponds to selecting a derived value.

Example 7.16. Continuing our example of conference registration, a user may fill in data as indicated in Example 7.15. He may then simply choose an operation “register”. However, the user may already be registered and simply want to change the number of additional banquet tickets from zero to one, in which he may choose an operation “change data”. In order to allow a user to enter only the data that really has to be changed, he could simply highlight the ‘tickets?’ field only, and e.g., dispense with re-entering an address or phone/fax number as indicated in Example 7.15.

Thus, in addition to a type we have to provide a set of operations, each of which would depend on some *selection type* indicating the parts of the presented data that must be selected. In addition, we may foresee additional input to be requested from the user, for which dialogue boxes could be used.

Example 7.17. Let us continue the conference registration example. Here the type t in Example 7.14 becomes the content data type $\text{cont}(\text{REGISTRATION})$ of an interaction type REGISTRATION . Note that in this case $t = t_{\text{REGISTRATION}}$ holds, because no links appear within the content data type. The query $q_{\text{REGISTRATION}}$ is defined as a constant query, i.e., it has the form $v : t$, in which the complex value v of type t represents the empty form as defined in Example 7.15. Whatever an underlying database looks like, executing the query results in a single interaction object (u, v) with this value and a generated value u of type URL .

Operations associated with this interaction type can be “change_data” and “register”. For the latter one the selection type would be

$$\begin{aligned} & (\text{first_name} : (\text{filled_in} : \text{STRING}), \text{last_name} : (\text{filled_in} : \text{STRING}), \\ & \quad \text{affiliation} : (\text{filled_in} : \text{STRING}), \text{address} : \\ & \quad (\text{filled_in} : \text{STRING}), \text{phone} : (\text{filled_in} : \text{PHONE}), \\ & \quad \text{fax} : (\text{filled_in} : \text{PHONE}) \oplus (\text{empty} : \mathbb{1}), \\ & \quad \text{email} : (\text{filled_in} : \text{EMAIL}), \text{student?} : (\text{filled_in} : \{\mathbb{1}\}), \\ & \quad \text{papers} : \{\text{INT}\}, \text{proceedings?} : (\text{filled_in} : \text{NAT}), \\ & \quad \text{tickets?} : (\text{filled_in} : \text{NAT})) \end{aligned}$$

which indicates that only the fax number is optional, but all other fields must be filled in. For the operation “change_data” the selection type could be

$$\begin{aligned} & (pt : (\text{proceedings?} : (\text{filled_in} : \text{NAT}), \text{tickets?} : \\ & \quad (\text{filled_in} : \text{NAT}))) \oplus (p : (\text{proceedings?} : (\text{filled_in} : \text{NAT})) \\ & \quad \oplus (t : (\text{tickets?} : (\text{filled_in} : \text{NAT})))) \end{aligned}$$

indicating that the number of extra proceedings or the number of additional banquet tickets or both must be highlighted – of course, we could foresee more options for change, but for the sake of brevity we omit these here.

This is again similar to the form-based approach, though the concept of operation is slightly more abstract than in the form-based approach.

Finally, we like to point out that web interaction types are in fact much more powerful than any forms model. They can be used to define query forms as consumption types or answer forms as production types. Furthermore, operations on web interaction types enable reasoning about WISs in higher-order dynamic logic. However, none of the extensions that we will discuss in the next sections apply to form-based approaches.

Furthermore, knowing which input is required by the operations associated with a media type allows a designer to develop a “real” form as part of the implementation of interaction types. This can be done semi-automatically on the basis of preferences for layout and playout style and the types used in the definition of the operation signatures. In this way low level forms processing becomes a generic system component without any need to be specified.

7.2 Adaptivity

In this section we start with the first of several extensions that will turn interaction types into web interaction types. This first extension is dedicated to adaptivity with respect to end-devices, communication channels and user preferences. More precisely, we concentrate on restrictions that require a modification of the presentation in the sense that only part of the information can be presented to the user. For instance, such restrictions may intrinsically result from the chosen end-device, e.g., in case of small, hand-held devices, or communication channels, e.g., in cases of restricted band-width. Furthermore, users may explicitly request that not all information represented by an interaction object shall be presented at once. Thus, the problem is to find relaxed ways of representing the data content of interaction objects.

The extension we will discuss in this section concentrates mainly on the automatic splitting of the data content. That is, an interaction object shall be split into a sequence of interaction objects, the combined content of which is equivalent to the content of the original interaction object. The interaction objects in such a sequence will provide links to the successor, so that sequentialisation operations can be used to navigate along the objects of the sequence to obtain the complete data content. Furthermore, the splitting shall be performed in such a way that the objects in the sequence provide data content of decreasing importance to the user, such that the navigation sequence can be quitted as soon as sufficient information has been retrieved to execute one of the operations associated with the interaction object.

We will discuss two alternatives for automatic splitting based on cohesion and proximity. This automatic splitting then realises the adaptivity to the

given restrictions. Only the restrictions as such have to be specified. In case of user preferences this is done as part of the user profile, whereas data about device and channel restrictions can be obtained at run-time from the user.

An alternative to splitting is aggregation, i.e., the data is compacted into a more coarse-grained representation. For this we provide aggregation functions, which we will discuss in the context of the impact of splitting of data content to the associated operations.

7.2.1 Cohesion Preorders

Cohesion introduces a controlled form of information loss. For this we exploit the lattice structure on content data types that results from subtyping. Formally, we define a partial order \leq on structure expressions:

- For any expression exp we have $exp \leq exp$ and $exp \leq \mathbb{1}$;
- For record structure expressions we have $(a_1 : exp_1, \dots, a_m : exp_m) \leq (a_{\sigma(1)} : exp'_{\sigma(1)}, \dots, a_{\sigma(n)} : exp'_{\sigma(n)})$ with injective $\sigma : \{1, \dots, n\} \rightarrow \{1, \dots, m\}$ and $exp_{\sigma(i)} \leq exp'_{\sigma(i)}$;
- For list, multiset and set structure expressions we have $\{exp\} \leq \{exp'\}$, $\langle exp \rangle \leq \langle exp' \rangle$, and $[exp] \leq [exp']$, respectively, iff $exp \leq exp'$ holds.

Naturally, if instead of the structure expression $cont(M)$ of an interaction type M we take the content data type t_M , and we have $cont(M) \leq cont(M')$, then we also have $t_M \leq t_{M'}$, i.e., the relationship between the content data types is given by subtyping.

Definition 7.18. If $cont(M)$ is the content structure expression of an interaction type M and $sup(cont(M))$ is the set of all content structure expressions exp with $cont(M) \leq exp$, then a preorder \preceq_M on $sup(cont(M))$ extending the order \leq on content expressions is called a *cohesion preorder*.

Small elements in $sup(cont(M))$ with respect to \preceq_M define information to be kept together, if possible. Clearly, $cont(M)$ is minimal with respect to \preceq_M . This enables a controlled form of information decomposition. If we want to decompose an interaction type or if we are forced to decompose according to user requirements or technical restrictions, then we may choose a minimal element $t_1 \in sup(cont(M))$ with respect to \preceq_M such that it satisfies the representation requirements. Note that if we only provide a preorder, not an order, then there may be more than one such t_1 .

Taking just t_1 instead of $cont(M)$ means that some information is lost, but this only refers to the first data transfer. When transferring data of type t_1 , we must include a link to a possible successor containing detailed information. In order to determine such a successor we can continue looking at all content types $t' \in sup(cont(M))$ with $t_1 \not\preceq_M t'$. These are just those containing the complimentary information that was lost. Again we can choose a minimal

type t_2 among these t' with respect to \preceq_M that requires not more than the available capacity. t_2 would be the desired successor.

Proceeding this way the whole communication is broken down into a sequence of suitable units t_1, t_2, \dots, t_n that together contain the information provided by the interaction type. Of course, the cohesion pre-order suggests that the relevance of the information decreases, while progressing with this sequence. The user may decide at any time that the level of detail provided by the sequence t_1, \dots, t_i is already sufficient for his or her needs.

Example 7.19. Let us ignore the defining query as well as operations. Just take the following simplified content data type of a media type HOME_LOAN:

$$(\text{type} : \text{STRING}, \text{conditions} : \text{STRING}, \text{interest} : \{ (\text{amount} : \text{CARD}, \text{rate} : \text{FLOAT}) \}).$$

So we could have the following instance:

$$(\&o_{11}, (\text{type} : \text{"Flexi"}, \text{conditions} : \text{"up to \$10,000 for 12 to 36 months"}, \text{interest} : \{ (\text{amount} : 3,000, \text{rate} : 8.2), (\text{amount} : 5,000, \text{rate} : 7.6), (\text{amount} : 10,000, \text{rate} : 7.3) \})).$$

We used the convention to denote URLs with a starting &. So, the URL for this interaction object is $\&o_{11}$. Neglecting the inner set type we could define a cohesion order (not just a preorder) by

$$\begin{aligned} &(\text{type} : \dots, \text{conditions} : \dots, \text{interest} : \dots) \\ &\quad \preceq (\text{type} : \dots, \text{conditions} : \dots) \\ &\quad \preceq (\text{type} : \dots, \text{interest} : \dots) \\ &\quad \preceq (\text{conditions} : \dots, \text{interest} : \dots) \\ &\quad \preceq (\text{type} : \dots) \\ &\quad \preceq (\text{conditions} : \dots) \\ &\quad \preceq (\text{interest} : \dots) \end{aligned}$$

Then assume that we can at most transfer $t_1 = (\text{type} : \dots, \text{conditions} : \dots)$. This would leave us with

$$\begin{aligned} &(\text{type} : \dots, \text{conditions} : \dots, \text{interest} : \dots) \\ &\quad \preceq (\text{type} : \dots, \text{interest} : \dots) \\ &\quad \preceq (\text{conditions} : \dots, \text{interest} : \dots) \\ &\quad \preceq (\text{interest} : \dots) \end{aligned}$$

Thus, the first part of the information that we transfer for the media object above would be

$$(\&o_{11}, (\text{type} : \text{"Flexi"}, \text{conditions} : \text{"up to ..."}, \text{more} : \&o_{11}^1)).$$

Next, we could choose $t_2 = (\text{type} : \dots, \text{interest} : \dots)$. The second bit of transferred information of the media object above would be

$$(\& o_{11}^1, (\text{type} : \text{“Flexi”}, \text{interest} : \{(\text{amount} : 3,000, \text{rate} : 8.2), \\ (\text{amount} : 5,000, \text{rate} : 7.6), (\text{amount} : 10,000, \text{rate} : 7.3) \})) .$$

In fact, we could stop here, as further processing would not lead to more information. We would be left with just

$$(\text{type} : \dots, \text{conditions} : \dots, \text{interest} : \dots) \preceq (\text{conditions} : \dots, \text{interest} : \dots)$$

Here, all maximal elements in $\text{sup}(\text{cont}(M))$ have disappeared, which indicates that no information has been left out.

7.2.2 Proximity Values

An alternative to cohesion preorders is to use *proximity values*.

Definition 7.20. Let \exp_1, \dots, \exp_n be an antichain with respect to \preceq . A symmetric $(n \times n)$ -matrix $\{p_{ij}\}_{1 \leq i,j \leq n}$ with $0 \leq p_{ij} \leq 1$ is called a *set of proximity values*.

The antichain in the definition represents a possible split of the information content. The higher the proximity value, the more do we wish to keep the components together.

Applying proximity values $\{p_{ij} \mid 1 \leq i, j \leq n\}$ requires also to determine first the maximum amount of data that should be transferred. Then for each $X \subseteq \{1, \dots, n\}$ determine its *weight*, i.e.,

$$w(X) = \sum_{i,j \in X, i < j} p_{i,j}$$

and its *greatest common subtype* $\text{gcs}(X)$, i.e., the greatest element $t_1 \in \text{sup}(\text{cont}(M))$ with $t_1 \leq \exp_i$ for all $i \in X$. We choose the X with largest weight such that the $\text{gcs}(X)$ satisfies the representation requirements. Proceeding this way we also construct a sequence of content types t_1, \dots, t_n , all appearing in the chosen anti-chain, such that together they provide the information of the media type. Same as for cohesion preorders the relevance of the information decreases, while progressing with this sequence, and the user may decide to stop the transfer, after receiving t_1, \dots, t_i .

Example 7.21. Let us take the same interaction type as in Example 7.19. We choose the antichain $\exp_1 = (\text{type} : \dots)$, $\exp_2 = (\text{conditions} : \dots)$ and $\exp_3 = (\text{interest} : \dots)$ and the proximity values $p_{1,2} = 0.8$, $p_{1,3} = 0.5$ and $p_{2,3} = 0.1$. Then we get the following weights and greatest common subtypes:

X	$w(X)$	$gcs(X)$
{ 1 }	0	(type : ...)
{ 2 }	0	(conditions : ...)
{ 3 }	0	(interest : ...)
{ 1, 2 }	0.8	(type : ..., conditions : ...)
{ 1, 3 }	0.5	(type : ..., interest : ...)
{ 2, 3 }	0.1	(conditions : ..., interest : ...)
{ 1, 2, 3 }	1.4	(type : ..., conditions : ..., interest : ...)

Assuming that the whole information, i.e., $gcs(\{2, 3\})$ has to be decomposed, the result would be the same sequence of types as in the previous example.

There is no general preference for cohesion preorders or proximity values. The major difference is that the proximity values provide an information split that is defined a priori, whereas the use of a cohesion (pre-)order would determine such a split. This means that cohesion (pre-)orders are more flexible for the price of being more costly with respect to the determination of the split.

7.2.3 Adaptation of Operations

Both cohesion and proximity values define sequences of interaction objects that are linked together. Following these links “more” leads to sequentialisation operations *follow* that are defined for the interaction type. These have already been discussed earlier in this chapter.

Most importantly, the splitting of an interaction object of type M into a sequence of interaction objects of types M_1, \dots, M_n , respectively, requires an adaptation of the operations associated with M . In general, we can assume that an operation O associated with M is also associated with at least one of the types M_i , but it could even be associated with all these interaction types. As data of the input types for O have to be selected from the user using dialogue boxes and the like, the crucial problem is the collection of data of the selection type $sel(O)$, as these may now be spread over several interaction objects.

Let us assume that operation O is associated with the type $M_k (k \in \{1, \dots, n\})$ resulting from splitting with either cohesion pre-order \preceq or proximity values $\{p_{i,j}\} (i, j \in \{1, \dots, n\})$. Let us further assume that t_i is the content data type of M_i for $i = 1, \dots, n$, while t_M is the content data type of M . Then $sel(O)$ is a supertype of t_M , and data of this type must be selected by the user to execute O . As $sel(O)$ is in general not a supertype of t_k , the user cannot select the required data from the value of type t_k that is presented to him or her.

Formally, as supertypes of t_M form a lattice with t_M as smallest and $\mathbb{1}$ as largest element and both $sel(O)$ and t_k are elements of this lattice, we can build their join $sel(O) \sqcup t_k$, which is the least common supertype of $sel(O)$

and t_k . Let us call this type $sel_k(O)$, as a user can select a value of this type from the data content presented to him or her by means of an interaction object of type M_k . So, the first requirement is that for the operation O on M_k the selection type is modified to $sel_k(O)$. This is still insufficient, but we can exploit that $sel(O)$ determines another minimal supertype \bar{t}_k of t_M , such that the meet of $sel_k(O)$ with this type \bar{t}_k is $sel(O)$.⁶

So, in summary, we build types $sel_k(O)$ and \bar{t}_k satisfying

$$\begin{aligned} sel_k(O) &= sel(O) \sqcup t_k \quad \text{and} \\ sel(O) &= sel_k(O) \sqcap \bar{t}_k \end{aligned}$$

with \sqcup and \sqcap denoting the usual join and meet operations in a lattice, respectively. The type $sel_k(O)$ becomes the selection type $sel(O)$ for the operation O on the interaction type M_k , but for O to become executable, additional input of type \bar{t}_k is needed. This input can be obtained in different ways:

- The input signature for O on M_k can be extended by $\iota_0 :: \bar{t}_k$. In this case we call the selection method for O in case of adaptivity *input extension*.
- The types $sel_i(O) = sel(O) \sqcup t_i$ with the content data types t_i of M_i for $i = 1, \dots, k-1$ define a sequence of selection types for O on previously visited interaction objects. These interaction objects may be kept open, and the user can be requested to select data on each of these interaction objects presented to him. Then $t'_k = \prod_{i=1}^k sel_i(O)$ defines the type of the collected data selected this way, and the complement $\bar{t}'_k = \prod\{t \mid sel(O) \leq t \sqcap t'_k\}$ defines the complement. Then the input signature for O on M_k can be extended by $\iota_0 :: \bar{t}'_k$. In this case we call the selection method for O in case of adaptivity *sequential selection*.
- Alternatively to sequential selection a join operation on the already visited interaction objects of types M_1, \dots, M_{k-1} can be performed first. Then the user can be requested to select a value of type $sel_{join}(O) = t'_k$ from the value presented to the user by means of the interaction object resulting from this join. In addition, the input signature for O on M_k must again be extended by $\iota_0 :: \bar{t}'_k$. In this case we call the selection method for O in case of adaptivity *join selection*.

As a result of our discussion we can state that in case a cohesion pre-order or proximity values have been defined for an interaction type M , then for each operation O on M the selection method in case of adaptation has to be defined as well. This adaptation method is one of input extension (E), sequential selection (S) or join selection (J).

7.2.4 Adaptivity by Means of Aggregation Operations

Let us finally discuss briefly an alternative to achieving adaptivity by means of aggregation operations instead of splitting and sequentialisation. In this

⁶ Of course, we have $\bar{t}_k = \prod\{t \mid sel(O) \leq t \sqcap sel_k(O)\}$, which is known as the complement of $sel(O)$ and $sel_k(O)$

case several aggregation operations O_1, \dots, O_n are required such that the resulting interaction types M_1, \dots, M_n have content data types t_1, \dots, t_n with $t_M = t_n \leq \dots \leq t_1$. In case of a need for adaptation due to restrictions an interaction object of type M_k would be built instead of an interaction object of type M , where k is largest such that the needed value of type t_k satisfies the restricting conditions.

Then we first create an interaction object of type M_k capturing the most important part of the data content. In addition to the value of type t_k we also present a link to an interaction object of type M_{k+1} , which would be presented next, if the user decides to follow this link to obtain a more complete representation. This continues in the same way with M_{k+2} and so on until the user reaches $M_n = M$. The user may of course also decide to quit this sequential presentation.

When this sequence of interaction objects of types M_k, M_{k+1}, \dots, M_n is to be created, the necessary aggregation operation may be executed at the server site hosting the underlying database. That is, first only a value of type t_k is to be created, transmitted and presented in order to cope with device and channel restrictions. For the creation of the follow-on value of the value of type t_i , i.e., a value of type t_{i+1} , the computation can be optimised in that only a value of the complement t'_i of t_{i+1} and t_i has to be created and transmitted. We have

$$t'_i = \bigcap \{t \mid t_{i+1} \leq t \sqcap t_i\}.$$

This means that $t_{i+1} = t_i \sqcap t'_i$, so that the data content of type t_{i+1} to be presented to the user can be obtained by a join operation using the already presented value of type t_i and the newly created value of type t'_i . This guarantees that only a sufficiently small portion of the data content of M is created at a time.

Thus, in case of the aggregation approach to adaptivity we also obtain sequentialisation operations, and there is a need to adopt the selection types of operations. However, for an operation O with selection type $sel(O)$ there is a smallest k such that $t_k \leq sel(O)$. Therefore, an option is to associate such an operation O only with the interaction types M_k, M_{k+1}, \dots, M_n . Then no change to the operation is required. In this case we call the modus *execution deferral*.

On the other hand, sequential selection and join selection cannot be applied, as previously presented values are values of supertypes that do not contain additional information to the one currently presented. Therefore, in case a sequence of aggregation operations has been defined for an interaction type M for the purpose of supporting adaptivity, then for each operation O on M the adaptation method has to be defined as well. This adaptation method is one of execution deferral (D) or input extension (E).

7.2.5 Adaptivity Extension

Let us finally summarise the discussion in this section on adaptivity.

Definition 7.22. Let $M = (cont(M), q_M, O(M))$ be an interaction type. An *adaptivity extension* of M consists of either

- a cohesion pre-order \preceq on $sup(cont(M))$, or
- an anti-chain exp_1, \dots, exp_n of expressions $exp_i \in sup(cont(M))$, each associated with a subset $O_i(M) \subseteq O(M)$ of the set of operations of M , together with proximity values $\{p_{i,j}\}_{1 \leq i,j \leq n}$, or
- a sequence of aggregation operations O_1, \dots, O_n with associated aggregation operators $op_i : t_M \rightarrow t_i$ such that $t_m \leq t_n \leq \dots \leq t_1$ holds.

In the case of anti-chains or proximity values each operation $o \in O(M)$ is associated with an adaptation method, which is either input extension (E), sequential selection (S) or join selection (J).

In case of aggregation operations each operation $o \in O(M)$ is associated with an adaptation method, which is either execution deferral (D) or input extension (E).

7.3 Granularity

Another possibility to tailor the information content of interaction types is to consider different versions that permit the user to express preferences with respect to finer- or coarser-grained presentations including the possibility to easily switch between these versions. This is similar to dimension hierarchies as in OLAP systems: moving down a hierarchy results in information growth, moving up in information loss. Such a hierarchy is already implicitly defined by the component or link structures, respectively.

7.3.1 Hierarchical Versions

Let $cont(M)$ be the content structure expression of an interaction type M and let $S(M) = sup(cont(M))$ be the lattice of content structure expressions exp with $cont(M) \leq exp$. For each $E \in S(M)$ we can define several equivalent versions that result from flattening links.

On the grounds of the Higher-order Entity Relationship Model (HERM) flattening can be formally defined by operators $flat_r$. If E' is a component of E corresponding to the role r , we may replace E by $flat_r(E)$ defined as follows:

$$\begin{aligned} comp(flat_r(E)) &= comp(E) - \{r : E'\} \cup comp(E') \text{ and} \\ attr(flat_r(E)) &= attr(E) \cup attr(E') \end{aligned}$$

The new key is $\text{id}(E) - \{r : E'\} \cup \text{id}(E')$ in the case $(r : E') \in \text{id}(E)$; it is $\text{id}(E)$ otherwise. We may extend this definition and flatten occurrences of links $\ell : M'$ in content data types $\text{cont}(M')$. We simply substitute $\text{cont}(M')$ for $\ell : M'$. The resulting interaction type will be denoted as $\text{flat}_\ell(M)$.

The converse operator raise_P for $P \subseteq \text{comp}(E) \cup \text{attr}(E)$ is defined analogously. Any subset $P \subseteq \text{comp}(E) \cup \text{attr}(E)$ allows to replace E by $\text{raise}_P(E)$ and a new database type E_{new} defined as follows:

$$\begin{aligned}\text{comp}(\text{raise}_P(E)) &= \text{comp}(E) - P \cup \{r_{\text{new}} : E_{\text{new}}\} , \\ \text{attr}(\text{raise}_P(E)) &= \text{attr}(E) - P , \\ \text{id}(\text{raise}_P(E)) &= \begin{cases} \text{id}(E) - P \cup \{r_{\text{new}} : E_{\text{new}}\} & \text{for } P \cap \text{id}(E) \neq \emptyset \\ \text{id}(E) & \text{else} \end{cases} \\ \text{comp}(E_{\text{new}}) &= P \cap \text{comp}(E) , \\ \text{attr}(E_{\text{new}}) &= P \cap \text{attr}(E) \quad \text{and} \\ \text{id}(E_{\text{new}}) &= \begin{cases} \text{id}(E) & \text{for } \text{id}(E) \subseteq P \\ P & \text{else} \end{cases} .\end{aligned}$$

Again, this may be generalized to $\text{raise}_{\text{exp}}(M)$ for a content expression occurring within $\text{cont}(M)$. This will introduce a new link expression replacing exp .

Flattening and raising can be defined in an analogous way for any content structure expression. Thus, for an expression $E \in \mathcal{S}(M)$ we define $[E]$ as the set of all content structure expressions that result from E by applying a sequence of flat-operations or their converses. Obviously we have $[E] = [E']$ iff $E' \in [E]$ holds.

Definition 7.23. Let M be an interaction type. Choose a set $\bar{H}(M)$ of content structure expressions such that for each $E \in \mathcal{S}(M)$ there exists a unique representative $E' \in [E]$ such that $E' \in \bar{H}(M)$ holds. A *set of hierarchical versions* of M is a finite subset $H(M)$ of $\bar{H}(M)$ with $M \in H(M)$.

Each cohesion pre-order \preceq_M on M induces a cohesion pre-order $\preceq_{M'}$ on each element $M' \in H(M)$.

Example 7.24. Let us look again at the scene INCOME and the action enter_income_details. The data content expression is

$$\text{cont}(\text{INCOME}) = (\text{requested_loan} : t_{ql}, \text{customer_data} : t_{qc}, \text{income_data} : t_{qi})$$

with

$$\begin{aligned}t_{ql} &= (\text{id} : ID, \text{amount} : CARD, \text{details} : LOAN_DETAILS), \\ t_{qc} &= (\text{customer_no} : CARD, \text{name} : STRING)\end{aligned}$$

and

$$t_{qi} = (\text{reg_income : } FLOAT, \text{reg_expenditure : } FLOAT, \\ \text{accounts : } \{\text{(account_no : } CARD, \text{institution : } STRING)\}).$$

Define $E_0 = \text{cont}(\text{INCOME})$,

$$E_1 = (\text{requested_loan : } (\text{id : } ID, \text{amount : } CARD), \text{customer_name : } STRING, \\ \text{reg_income : } FLOAT, \text{reg_expenditure : } FLOAT),$$

$$E_{21} = (\text{loan_amount : } CARD, \text{customer_name : } STRING, \\ \text{reg_income : } FLOAT)$$

and

$$E_{22} = (\text{loan_amount : } CARD, \text{reg_income : } FLOAT, \\ \text{reg_expenditure : } FLOAT).$$

Then $H(M) = \{E_0, E_1, E_{21}, E_{22}\}$ is a set of hierarchical versions of M .

As indicated in Section 7.2 a user can switch between hierarchical versions by means of generalisation and specialisation operations. Let these be called *up* and *down*, respectively. In both cases the selection type as well as the input and output types are $\mathbb{1}$, i.e., nothing is to be selected, no additional input is required, and no output is produced.

The body of *up* is simply $\text{OPEN } M_{i+1}(val = \pi_{i,i+1}(v_i))$. Here v_i is the value of type t_i associated with $E_i \in H(M)$ that is currently presented to the user, M_{i+1} is the interaction type defined by E_{i+1} such that $E_i < E_{i+1}$, but there is no j with $E_i < E_j < E_{i+1}$, and $\pi_{i,i+1}$ is the canonical projection from t_i to t_{i+1} . Note that in this case E_{i+1} is not uniquely determined, so an interaction with the user to select the appropriate coarser-grained version is required.

Analogously, the body of *down* is $\text{OPEN } M_{i-1}(val = \pi_{i-1}(v))$. Here v is the value of type t_0 associated with $E_0 \in H(M)$ and π_{i-1} is the canonical projection from t_0 to t_{i-1} – so we have that $\pi_i(v)$ is the value that is currently presented to the user. M_{i-1} is the interaction type defined by E_{i-1} such that $E_{i-1} < E_i$, but there is no j with $E_{i-1} < E_j < E_i$.

7.3.2 Adaptation of Operations

A set $H(M)$ of hierarchical versions defines also hierarchical versions for each interaction object o of type M . For $H(M) = \{E_0, \dots, E_n\}$ we obtain versions o_0, \dots, o_n of types M_0, \dots, M_n , where M_i corresponds to E_i . In particular, we have $M_0 = M$ and $o_0 = o$, so each o_i results from o_0 by means of the canonical projection defined by $\text{cont}(M) \leq \exp_i$ with $E_i \in [\exp_i]$.

As each operation $O \in O(M)$ is associated with o_0 , we are free to associate with each $E_i \in H(M)$ any subset $O_i(M) \subseteq O(M)$. As for the adaptation of operations in case of cohesion or proximity values the key question is the definition of the adaptation method, by means of which the value of the selection type $\text{sel}(O)$ can be determined for $O \in O_i(M)$.

For this, we build again types $sel_i(O)$ and \bar{t}_i satisfying

$$\begin{aligned} sel_i(O) &= sel(O) \sqcup t_i \quad \text{and} \\ sel(O) &= sel_i(O) \sqcap \bar{t}_i . \end{aligned}$$

The type $sel_i(O)$ becomes the selection type for the operation $O \in O_i(M)$, but for O to become executable, additional input of type \bar{t}_i is needed. This input can be obtained in the similar ways as discussed in Section 7.2.

input extension (E): The input signature for $O \in O_i(M)$ can be extended by $t_s :: \bar{t}_i$, i.e., the missing input is requested from the user.

specialisation (SP): Choose a maximal $E_j \in H(M)$ such that $exp_j < exp_i$ (for $exp_i, exp_j \in sup(cont(M))$ with $E_j \in [exp_j]$ and $E_i \in [exp_i]$) and $exp_j \leq sel(O)$ hold. In this case the interaction object o_j can be reached from o_i by means of a specialisation operation, and the selection becomes possible on o_j .

Let us finally summarise the discussion in this section on granularity.

Definition 7.25. Let $M = (cont(M), q_M, O(M))$ be an interaction type. An *extension* of M by hierarchical versions consists of a set of hierarchical versions $H(M) = \{E_0, \dots, E_n\} \subseteq \bar{H}(M)$ together with sets of operations $O_i(M)$ for $i = 0, \dots, n$. In particular $E_0 = cont(M)$ and $O_0(M) = O(M)$. Each operation $o \in O_i(M)$ is associated with an adaptation method, which is either input extension (E) or specialisation (SP).

7.4 Web Interaction Schemata

Let us first summarise the extensions to interaction types, which give us the definition of a web interaction type.

Definition 7.26. A *web interaction type* (WIT) M is an interaction type $(M, cont(M), q_M)$ together with a set $O(M)$ of operations (including the generic operations) on the interaction type M , an adaptivity extension $\mathcal{A}(M)$ for M , and an extension $\mathcal{H}(M)$ by hierarchical versions.

Let us write

$$M = (cont(M), q_M, O(M), \mathcal{A}(M), \mathcal{H}(M))$$

for a web interaction type with name M and all the extensions in Definition 7.26.

A *web interaction schema* is a finite closed set \mathcal{S} of web interaction types, i.e., it extends the content schema defined by the underlying interaction types. Consequently, a database over the underlying database schema defines first a pre-site, i.e., a set of interaction objects (u, v) , where u is a URL and v is a

value of type t_M for each $M \in \mathcal{S}$. Note that these values v are linked to each other, as they contain values of type *URL*.

Each extension in Definition 7.26 now defines extensions to the pre-site. An interaction object together with all its extensions will be called a *web interaction object*, and the set of all such web interaction objects extend the pre-site to a *site*.

First, all operations in $O(M)$ are associated with all interaction objects (u, v) of type M , i.e., if a value of type $\text{sel}(O)$, a supertype of t_M , is selected from v , then operation O can be initiated on the given interaction object, collecting additional input for the input types of O , producing output for the output types of O , and executing the body of O .

The adaptivity extension defines modifications of the interaction object (u, v) , i.e., a sequence of variant interaction objects $(u_1, v_1), \dots, (u_n, v_n)$ that are linked to each other. That is, a user expecting to obtain (u, v) and the operations $O(M)$ will be presented (u_1, v_1) instead, and $O(M)$ is also modified to a set $O_1(M)$ of operations on (u_1, v_1) . From here the user may navigate to (u_2, v_2) , (u_3, v_3) and so on until he or she has received sufficient information and executed the necessary operations. With respect to the story this means that the scene represented by (u, v) will be explored step-by-step using (u_1, v_1) , (u_2, v_2) , (u_3, v_3) and so on, and the execution of operations or the use of explicit navigation links may lead the user to the next scene.

The extension by hierarchical versions defines different versions for the interaction object (u, v) and all its variants (u_i, v_i) . Let $(u_i^1, v_i^1), \dots, (u_i^k, v_i^k)$ be the versions of the variant (u_i, v_i) . Then a user, instead of being presented (u_i, v_i) , will be confronted with the default version, but he or she can navigate through the hierarchy to any of the other versions using the operations *up* and *down*. Furthermore, on the version (u_i^j, v_i^j) the set of available operations $O_i(M)$ is replaced by the operations $O_i^j(M)$ as discussed for the adaptation of operations in case of the presence of hierarchical versions.

In the next chapter in Definition 8.7 we will further extend the notions of web interaction type and web interaction schema.

7.5 Bibliographical Remarks

The work in [222] presents the forerunner of the theory of *interaction types* (see, e.g., for media types [526, 720, 708, 709] and for dialogue objects [703, 716]). Interaction types provide a powerful means for derivation of queries from keywords and thus can be combined with approaches such as [100, 218, 267, 290, 403]. The entity-relationship model has been introduced by P. P. Chen in 1976 [146] as a generalisation of the network model formalisation of C. Bachman [50, 51]. The model conceptualises and graphically represents structuring of the relational model and is currently used as the main conceptual model. Most tools supporting information systems development use UML [211, 538, 918] or an enhanced entity-relationship model (EER)

[209, 279, 299, 328, 790]. The enhanced entity-relationship model is the topic of the book [823]. Brief surveys are [838, 840].

We advocated the local-as-view approach [586, 741]. Data-intensive services are supported by various database systems. Views are defined on the basis of classical database technology. Some of them may be materialised. Queries are also issued against these views. A view has its own schema [360, 501, 715] and uses database federation approaches [932]. SQL and database technology consider one-table views. We realised that WIS management is better supported by complex views which are defined as a database scheme.

Furthermore, our approach enhances view technology by explicit integration of functionality. This extension uses results achieved for object-oriented database enhancements [221, 704, 705, 717, 718, 890, 891]. Object-orientation started with an integrated approach to structuring and functionality – at least for each of the classes. This integrated approach can be generalised to views that are supported by functions such as aggregation, generalisation, specialisation, reordering, browsing sequentialisation, join, link, search, presentation, and visualisation. These functions can be defined as generic operations based on structural recursion [108, 515, 546, 653, 692].

Structural recursion is a powerful concept for programming [68, 153, 342, 810, 118] and sufficient for object-relational databases [704, 705, 823]. We also used classical concepts developed for programming languages such as principles of programming, e.g., [193, 320], rational trees, e.g., [412, 540], compiler construction, e.g., [651, 921], type systems and reflection, e.g., [171, 705], general technology of object-oriented programming, e.g., [566], conceptual modelling techniques, e.g., [119], artificial intelligence programming techniques, e.g., [272, 691, 925], and hypertext programming techniques, e.g., [199, 277, 555, 698].

Database technology uses generic, parametric or higher-order functions such as insert, delete, and update. These functions are defined as general set or multiset operations with a type as parameter. Parametric procedures have already been used for ALGOL 60 and ALGOL 68 as procedures that could use another function or procedure as a parameter [344, 408, 480, 481]. Earlier, assembly macros have already been using this technology. This concept is similar to the government and binding approach in linguistics [148, 149]. Generic functions [85, 86] can be specialised, enhanced by context or instantiated for some of its parameters.

Genericity is a more general solution than dynamic systems [5, 666]. Dynamic systems react to changes in their environment in a quick and flexible way. For instance, a workflow or system can choose the most appropriate flow of action at runtime according to the current values of some variables. Generic workflows [872, 873] have been applied for support of complex dynamically changing applications [872, 86, 871]. This approach can be generalised to techniques for generic structures and generic stories [370, 244, 363] as well as for representation forms [675].

Classical functions [277] for web pages can be defined as generic functions. These definitions can be given for input, output, search, navigation and other functions in dependence on their specific kind. For instance, we distinguish kinds of links such as embedded links, structural links (siblings, shortcuts, associative (might be of interest), selective/hierarchical linking, within-page links, outbound links, incoming links, avoiding tunnelling and splash screens [929].

Data spaces [446] and similarly visualisation spaces [261, 576] are specific functionality-enhanced interaction types. Interaction types provide content and functionality. Typical kinds of functionality are: navigation [236, 277], linking [84, 929], content enhancement [142, 121, 681], search techniques beyond querying [380, 204, 363], retrieval [202], metadata enhancements by dockets [458, 755], filtering and methods of tailoring [720], export and import features [234, 363, 869, 882], privacy protection features [10, 15, 16, 96, 847], reference propagation features references, XML and SGML techniques [2, 137, 280, 511, 920], XSLT-based facilities [416, 413, 425, 424], and integration facilities [525]. Database schema integration is an old issue that has attracted a lot of research [405, 435, 487, 493, 823, 861, 915]. Visualization may be based on Visual SQL [356]. It uses the entity-relationship model. At the same time, it has the same expressive power as SQL-92. The basics of cube technology have been developed in [497, 575].

Adaptive and flexible interfaces must cope with a variety of devices [4] and specific requirements such as security, privacy, and ethics [14, 96, 615]. A typical case is recommender systems [1, 82, 336, 383] with Adaptive Collaborative Filtering mechanisms. The exploration of user logs provides a rich source for web interaction types development, e.g., for navigation refinement [58, 292] or interface design [574].

Information is defined in the pragmatic sense [845, 438]. Users consume information provided by the WIS and deliver data to the WIS. Information may also be treated in the syntactic, semantic and pragmatistic sense. The anthropomorphic treatment is however the most appropriate for WIS.

Users collaborate within websites. Web 2.0 and the knowledge web challenge existing collaboration approaches which used, for instance, the 3C paradigm (communication, cooperation, coordination) [731]. Especially community websites must be supported by specific storyboarding and specific generic user models [852].

Forms generalise the classical menu-based presentation mechanisms [196, 675]. They can be enhanced to query-answer forms [72, 853, 528, 720] and input-out-forms [363, 364]. An enhancement is the W*H framework [175]. It can be easily combined with form-based reasoning [151], view enhancements [360, 897], and menu techniques (e.g., [53]).

Adaptivity is nowadays an area of intensive research due to the large variety of web users. Adaptivity and adaption of systems to users was already a topic in the late 1980s and in the 1990s (e.g., [212, 433, 470, 594]). The automatic adaptation of web pages to users, their environment and context,

their history of website usage, their communities, and their culture is still a challenging task [90, 250, 362, 406, 609, 667, 670, 898]. We use an approach that allows adaptation on the basis of parameter instantiation and automatic decomposition of content and functionality in dependence on coherence of content and functionality components. The decomposition of content is similar to the Venetian blind design for XML documents [424]. Objects are first represented with all relevant aspects. Their slicing into viewpoints depends on the complexity of the content, its inner structure, the adhesion and cohesion of the substructures, and the capacity of the interface that a user utilises. Modelling of decomposable units is similar to modelling of macromolecules used in structural chemistry (e.g., [62]) and structural biology (e.g., [817]). The background for our cohesion and adhesion approach lies in the theory of ligands [65, 198, 948] and the theory of components [828, 829, 846].

Key Messages

Interaction Types

- support *interface abstraction* for elementary scenes of the storyboard;
- combine *content* extracted from underlying databases and *operations* associated with the scene;
- support *direct navigation* by generated links between interaction objects;
- support *generic operations* for presentation, navigation and search.

Web Interaction Types provide an extension

- to support automatic *adaptivity* of content and functionality to end-devices, communication channels and user preferences;
- to support different *granularity* of content and associated functionality, and associated operations to switch between different versions.



Advanced Web Interaction Concepts ♣

This brief chapter contains further material for advanced reading, which readers with a primary interest in the methodology may skip. In Section 8.1 we discuss further extensions to web interaction types addressing measuring systems, ordering and contexts. In Section 8.2 we discuss the usage of web interaction types beyond pure interface abstraction, which was only sketched before in [738]. This will round up the integration of the conceptual model with the usage model of storyboarding.

Context integration was previously sketched in [401, 399]. The aspect of collaboration has been briefly dealt with in [91] and picked up again in [852]. The customisation of content has been initially discussed in [745]. Furthermore, the integration of web interaction types was investigated in [526]. In [532] the ideas underlying web interaction types were exploited for the definition of a model of web services.

8.1 Extensions to Web Interaction Types

In this section we will briefly discuss further extensions to interaction types concerning measuring, ordering, presentation and contextual information. The former three will impact the presentation to be discussed in Chapter 9. The extension by contextual information links interaction types with the various facets of contexts that were discussed in Section 5.4 in Chapter 5.

8.1.1 Measuring Systems and Ordering

Some of the base types used for the underlying database and thus also for the interaction types define sets of numerical values. We already used types such as *CARD*, *INT* and *FLOAT* in our examples. However, for a user values such as 7,000, 56 or -.392E2 only have a precise meaning when they are accompanied by a measuring unit, e.g., interpreting these values as 7,000 €, 56 km/h or

39.2°C , respectively, thus the values are used to represent an amount of money, a speed and a temperature, respectively.

A user may choose between different measuring units for the same purpose. For instance, an amount of money could be presented in any currency, a speed could be represented also in miles per hour, and a temperature could be given in ${}^\circ\text{F}$. When a system provides several such measuring units, we call a set of measuring units for the same purpose a *measuring system*.

Usually, in a measuring system one measuring unit is marked as the default, which is used, unless the specification of the user profile marks a different unit as the default.

If a measuring system $\{m_0, \dots, m_k\}$ is provided, then we must also foresee conversion functions that allow a user to switch from a presentation using one measuring unit m_i to another one using the measuring unit m_j . These are functions $c_{i,j} : b \rightarrow b$ for all $i, j \in \{0, \dots, k\}$ with $i \neq j$.

Example 8.1. Let $\{\text{km/h}, \text{mi/h}, \text{knot}\}$ be a measuring system for speed. Then we have conversion functions

$$c_{0,1} : \text{FLOAT} \rightarrow \text{FLOAT} \text{ with } s \mapsto \frac{1}{1.609344} s$$

converting a speed given in km/h into a speed given in mi/h, and

$$c_{2,0} : \text{FLOAT} \rightarrow \text{FLOAT} \text{ with } s \mapsto 1.852 s$$

converting a speed given in knots into a speed given in km/h.

Similarly, a conversion function for a temperature measuring system $\{{}^\circ\text{C}, {}^\circ\text{F}\}$ provides two conversion functions $c_{0,1}$ and $c_{1,0}$, both of type $\text{FLOAT} \rightarrow \text{FLOAT}$ with $t \mapsto \frac{9}{5}t + 32$ and $t \mapsto \frac{5}{9}(t - 32)$, respectively.

Note that a currency conversion would need to be defined by some service that is subject to changes.

This gives rise to measuring extensions for interaction types.

Definition 8.2. A *measuring extension* of an interaction type M assigns to each base type b in $\text{cont}(M)$ a measuring system comprising a set $\mu(b) = \{m_0, \dots, m_k\}$ of measuring units, among which m_0 is the default, together with conversion functions $c_{i,j} : b \rightarrow b$ for all $i, j \in \{0, \dots, k\}$ with $i \neq j$.

We may write $\mu(M) = \{(\mu(b), \gamma(\mu(b)))\}_b$ for a measuring extension of M , where $\gamma(\mu(b))$ denotes the set of conversion functions on $\mu(b)$.

While a measuring extension of an interaction type M refers to the base types used in $\text{cont}(M)$, an ordering extension refers to the bulk type constructors $[]$, $\{\cdot\}$ and $\langle \cdot \rangle$ for finite lists, sets and multisets, respectively. It simply provides ordering functions $o : [t] \rightarrow [t]$, $o : \{t\} \rightarrow [t]$ and $o : \langle t \rangle \rightarrow [t]$, i.e., all sets and multisets are also to be presented as lists, the latter ones with repetitions permitted.

An ordering function $o : bt \rightarrow [t]$ (where bt is one of $[t]$, $\{t\}$ or $\langle t \rangle$) is defined by a partial order $\leq : t \times t \rightarrow \text{BOOL}$ and structural recursion

$$o_{\leq}(X) = \text{src}[], \text{single}, \text{merge}_{\leq}]$$

with $[]$ denoting the empty list, single denoting the mapping $t \rightarrow [t]$ with $v \mapsto [v]$, and a merge function $\text{merge}_{\leq} : [t] \times [t] \rightarrow [t]$ that is defined as follows (here \circ denotes list concatenation):

$$\begin{aligned} \text{merge}_{\leq}([], Y) &= Y \\ \text{merge}_{\leq}(X, []) &= X \\ \text{merge}_{\leq}([v] \circ X, [w] \circ Y) &= \begin{cases} [v] \circ \text{merge}_{\leq}(X, [w] \circ Y) & \text{if } w \not\leq v \\ [w] \circ \text{merge}_{\leq}([v] \circ X, Y) & \text{if } w \leq v \end{cases} \end{aligned}$$

That is, a set of ordering functions is to be assigned to each occurrence of a bulk type constructor in $\text{cont}(M)$. Let these occurrences be captured by the set of indices $\text{ind}(M)$. Among such ordering functions we have again one default.

Definition 8.3. An *ordering extension* of an interaction type M assigns to each bulk type occurrence $in \in \text{ind}(M)$ a set $\text{ord}(in) = \{o_0, \dots, o_k\}$ of ordering functions, among which o_0 is the default, defined by partial orders \leq_i for $i = 0, \dots, k$, i.e., $o_i = o_{\leq_i}$.

In Section 7.1.4 we introduced reordering operations that are determined by an ordering extension. Let the operation name be *reorder*, or more precisely *reorder* $_{\leq}$ for some partial order \leq . Then there is no input nor output, i.e., the in- and output-types are both \mathbb{I} . However, the selection type $\text{sel}(\text{reorder}_{\leq})$ is a supertype t of $\text{cont}(M)$ that is determined by a bulk type occurrence $in \in \text{ind}(M)$. Then we must have $\leq = \leq_i$ for some ordering function $o_i \in \text{ord}(in)$. Thus, the body of $\text{sel}(\text{reorder})_{\leq}$ must be

$$\text{CLOSE} \parallel \text{OPEN}(\text{val} = \text{substitute}_{[in(v) \mapsto o_{\leq}(in(v))]}(v)) ,$$

where v is the actual interaction object of type M , $in(v)$ denotes the bulk value at the location in of v , and the function *substitute* replaces this bulk value by an ordered list $o_{\leq}(in(v))$, but otherwise leaves v unchanged.

8.1.2 Presentation Options

The structure expression $\text{cont}(M)$ of an interaction type M defines the data content of each of its interaction objects. A presentation option determines how a value of the associated content data type t_M is to be represented. In Chapter 2, Section 2.1 we already emphasised the importance of *ambience categories* such as energetic, romantic, elegant, refreshing, harmonic, stimulating, etc., which define the general impression a site is to give to its users.

Therefore, let \mathcal{A} be the finite set of all ambience categories that are to be supported. Choosing a subset $amb(M) \subseteq \mathcal{A}$ of ambience categories that are to be supported for the representation of values of type t_M is part of the presentation options.

Then the task is to assign to each base type and each constructor used in t_M a rule determining its representation. Such a *presentation rule* for a type t will determine for each base type b occurring in t how values of that type are presented to the user. This may comprise

- a format, e.g., for numbers to be presented,
- a decision, if the associated measuring unit is to be displayed as well, and
- a font including its size and display style (bold, slanted, italic, underlined, etc.).

Furthermore, a presentation rule for t determines for each type constructor appearing in t , how values of the involved types will be combined in a presentation of values of type t . This may comprise the following decisions:

- In case of the record type constructor, i.e., $t = (a_1 : t_1, \dots, a_n : t_n)$, the rule may involve the decision, if the field names a_i are to appear in the presentation of values of type t or not.
- In case of the list type constructor, i.e., $t = [t']$, the rule may involve the decision, if lists of values of type t' are to appear as dropdown lists or as lists of values together with checkboxes or radio buttons or the like.
- In case of the set or multiset type constructor, i.e., $t = \{t'\}$ or $t = \langle t' \rangle$, the rule may involve a decision how to map values of type t onto values of the list type $[t']$ – randomly or by means of a sorting criterion φ – plus the decision how to present lists of values of type t' .
- In case of the union type constructor, i.e., $t = (a_1 : t_1) \oplus \dots \oplus (a_n : t_n)$, the rule may involve the decision, if the field name a_i of a value $(a_i : v_i)$ is to appear in the presentation or not.

Definition 8.4. A *presentation extension* of an interaction type M consists of a set $amb(M) \subseteq \mathcal{A}$ of applicable ambience categories, a lossless decomposition of the type t_M into types t_1, \dots, t_k , and a presentation-options-mapping \mathcal{Pr} , which assigns to each pair (a, t_i) with $a \in amb(M)$ a set $\{r_{i1}^{(a)}, \dots, r_{in_i}^{(a)}\}$ of presentation rules for the type t_i . Furthermore, for each type t_i and ambience category a the presentation rule $r_{i1}^{(a)}$ is set as the default presentation rule.

Given a presentation extension for the interaction type M and an interaction object o of type M , then there is always an actual ambience $a \in amb(M)$ associated with o . Furthermore, o is decomposed into values o_i of type t_i for $i = 1, \dots, k$, so there are always actual presentation rules $r_{ij_i}^{(a)}$ (for $i = 1, \dots, k$, $1 \leq j_i \leq n_i$) associated with o . So $(a, r_{1j_1}^{(a)}, \dots, r_{kj_k}^{(a)})$ defines the actual presentation option for o .

Naturally, if o is projected onto a value o' of a supertype in $\text{sup}(\text{cont}(M))$ – this is the case for decompositions in case of adaptivity or for versions in case of hierarchical versions – then each decomposition of o into values o_1, \dots, o_k of types t_1, \dots, t_k gives rise to a decomposition of o' into values o'_1, \dots, o'_k of supertypes t'_1, \dots, t'_k . As all base types and constructors for t'_i also occur in t_i , the actual presentation option for o uniquely determines the presentation option for o' .

Finally, let us reconsider the generic presentation operations that we sketched earlier in this chapter. The purpose of these functions is either to change the measuring unit of some basic value appearing in the actually presented interaction object o , or to change the ambience category, or to change a presentation rule.

- In the first case let the operation name be *change_unit*. In this case the input type is the set of measuring units, and the selection type is a base type b appearing in t_M . There is no output, i.e., the output type is $\mathbb{1}$. If the input is $m_i \in \mu(b)$ and the actual measuring unit for values of type b in o is $m_c \in \mu(b)$, then the body of the operation replaces the actual value v_b of type b in o by $c_{c,i}(v_b)$ using the conversion function that is associated with m_c and m_i .
- In the second case let the operation name be *change_ambience*. In this case the input type is the set \mathcal{A} of ambience categories, and both the selection type and the output type are $\mathbb{1}$, i.e., nothing is to be selected and there will be no output. If the input is $a \in \text{amb}(M)$, then the result of *change_ambience* applied to an interaction object o of type M is the same as o except for the change of the ambience in the actual presentation option for o . If the current presentation option is $(b, r_{1j_1}^{(b)}, \dots, r_{kj_k}^{(b)})$, then the resulting presentation option will be $(a, r_{1\ell_1}^{(a)}, \dots, r_{k\ell_k}^{(a)})$, where $r_{i\ell_i}^{(a)} = r_{ij_i}^{(b)}$ – i.e., the presentation rule remains unchanged – if $r_{ij_i}^{(b)}$ is a presentation rule that is defined for ambience a and type t_i ; otherwise we will have $r_{i\ell_i}^{(a)} = r_{i1}^{(b)}$, i.e., the default presentation rule for values of type t_i under ambience a is taken.
- In the third case let the operation name be *change_presentation*. In this case the selection type is the union type $(\text{sel}_1 : t_1) \oplus \dots \oplus (\text{sel}_k : t_k)$ with the types t_1, \dots, t_k used for the decomposition of t_M , the input type is the set of names of presentation rules, and the output type is again $\mathbb{1}$. If the input r is one of the presentation rules $r_{i\ell}^{(a)}$ for the actual ambience category a , then the rule simply changes the current presentation option $(a, r_{1j_1}^{(a)}, \dots, r_{kj_k}^{(a)})$ for o by replacing $r_{ij_i}^{(a)}$ by r .

8.1.3 Contexts

In Section 5.4.1 we discussed contexts of web information systems, which led to a classification into the usage context comprising actor, storyboard, system

and temporal contexts, and the website context comprising provider, developer, organisational and social contexts. Among these the actor, storyboard and temporal contexts give rise to further extensions of interaction types. This will be elaborated in depth in connection with the navigation context in Chapter 8.2. Here we start with giving a glimpse of a contextual extension of interaction types.

The problem is to provide the information the user has already seen since entering the WIS in a condensed form to guide the navigation of the user through the system. Firstly, this depends on the type of user thus capturing characteristics from the actor context concerning additional information and functions required by the user. Secondly, it certainly depends on the history of the user's navigation through the system thus capturing characteristics from the storyboard context concerning the information the user needs to understand where he or she is and how he or she came there to decide on how to proceed with the story, i.e., the navigation through the WIS. Thirdly, it depends on the temporal context concerning the history of utilisation including possible interruptions.

A first attempt to conceptually capture all these aspects of contextual information by means of an extension of interaction types is to exploit contextual information bases (CIBs) [12, 818, 819] in generalised form. According to the theory of CIBs a context is a set of objects, each having several names, and each of these names may be coupled with a reference to another context. There may be names for objects that are not referencing other contexts. Here, the term “object” is used in the sense of “object identifier”, i.e., a unique abstract handle to identify objects.

Let us now bring together interaction types and contextual information bases. The obvious questions are:

- What are the objects that are required in contextual information bases, if we are given interaction types?
- What are the references that are required in contextual information bases?
- Is it sufficient to have names for describing objects in a context or should these be replaced by something else?

The natural idea for generalising the notion of object in contexts is to choose the interaction objects. Evaluating the defining query for an interaction type M leads to a set $\{(u_1, v_1), \dots, (u_n, v_n)\}$ of interaction objects. Recall that the u_i are values of type URL , whereas the v_i are values of the representing type t_M . As these URL-values are unique, they identify the interaction objects, and thus can be used as surrogates for them in the notion of context. This answers our first question. The objects are the interaction objects. The object identifiers needed in the contexts are the (abstract) URLs of these interaction objects.

However, one important aspect of interaction types is the use of classification abstraction. Conceptually, we do not define a set of interaction objects, but we generate them via queries defined on some underlying database schema.

Therefore, instead of the URLs of interaction objects we can exploit the names of the corresponding interaction types.

As we want to have access to path information, we may want to reference back to the various interaction objects a user has encountered so far. These are placed in several contexts, one of which is the right one corresponding to the user's path. However, we may also have different references, which lead to different contexts. So, the contexts we asked for in the second question are just the contexts for the interaction objects.

As to the third question, we definitely want to have more information than just a name. Fortunately, our theory of interaction types is already based on the assumption of an underlying type system. Thus, we simply have to replace the names by values of any type allowed by the type system.

Definition 8.5. A *context* C is a triplet (M_i, t_i, r_i) , where M_i is an identifier for an interaction type, t_i is a type, and r_i is either a reference $\rightarrow C'$ to a context C' or nil , the latter one indicating that there is no such reference.

We write $C = \{t_1 : M_1 \rightarrow C_1, \dots, t_\ell : M_\ell \rightarrow C_\ell\}$. If there is no reference for the i 'th name, i.e., we have (M_i, t_i, nil) , we simply omit $\rightarrow C_i$ and write $t_i : M_i$.

With such contexts the following functions **look-up** and **cross-ref** allow a user to traverse back a path in the story board and to explore alternative access paths, respectively.

- The function **look-up**(C, t) requires the name of a context C and a type t as input. It returns a sequence of values of types $t_i = t_i^0, \dots, t_i^{k_i} = t$ starting from context C and ending in t .
- The function **cross-ref**(p, C) requires a path $p = t^0, \dots, t^\ell$ of types and the name of a context C as input. It returns paths $t_i = t_i^0, \dots, t_i^{k_i}$ starting from context C and ending in the type specified by p , i.e., $t^\ell = n_i^{k_i}$.

Definition 8.6. A *contextual extension* of an interaction type M is a set of contexts $\{C^1, \dots, C^n\}$ with $C^i = \{t_1^i : M_1^i \rightarrow C_1^i, \dots, t_{\ell_i}^i : M_{\ell_i}^i \rightarrow C_{\ell_i}^i\}$ such that each M_j^k is the name of an interaction type, from which M can be reached by means of some action or navigation link, C_j^k is a context associated with M_j^k , and t_j^k is a supertype of the content type of M_j^k .

8.1.4 Extended Web Interaction Schemata

Let us first integrate the extensions into the Definition 7.26 of a web interaction type.

Definition 8.7. A *web interaction type* (WIT) M is an interaction type $(M, cont(M), q_M)$ together with a set $O(M)$ of operations (including the generic operations) on the interaction type M , an adaptivity extension $\mathcal{A}(M)$ for M , an extension $\mathcal{H}(M)$ by hierarchical versions, a measuring extension

$m(M)$, an ordering extension $o(M)$, a presentation extension $pr(M)$, and a contextual extension $C(M)$.

Let us write

$$M = (\text{cont}(M), q_M, O(M), \mathcal{A}(M), \mathcal{H}(M), m(M), o(M), pr(M), C(M))$$

for a web interaction type with name M and all the extensions in Definition 8.7.

A *web interaction schema* is still a finite closed set \mathcal{S} of web interaction types, and a database over the underlying database schema defines first a pre-site, i.e., a set of interaction objects (u, v) , where u is a URL and v is a value of type t_M for each $M \in \mathcal{S}$. An interaction object together with all its extensions will still be called a *web interaction object*, and the set of all such web interaction objects extend the pre-site to a *site*.

Each extension in Definition 8.7 defines extensions to the pre-site. For operations, the adaptivity extension and the extension by hierarchical versions this has already been discussed following Definition 7.26. The measuring extension defines the actual value v of an interaction object by means of the default measuring unit, and provides the conversion operations to switch to a different value. Analogously, the ordering extension defines the order of elements of a bulk type in the value v , and provides reordering operations. The presentation extension $pr(M)$ defines a default presentation of the value v of an interaction object, and provides operations to change units, ambience and presentation.

Finally, the contextual extension provides a set of contexts for M . Each context provides types, web interaction types and other contexts. By means of navigation the web interaction types appearing in these contexts give rise to the identifiers of previously visited web interaction objects that will be associated with the actual web interaction object (u, v) , and the types define values extracted from these web interaction objects that show the history in condensed form. Furthermore, the functions `look-up` and `cross-ref` are provided as additional operations (with trivial selection type \mathbb{I}) on each web interaction object.

8.2 Session Support, Navigation Contexts and Collaboration

In the previous chapter we emphasised web interaction types as means for interface abstraction in web information systems. In a nutshell, a web interaction type integrates data and functionality as well as adaptivity, granularity, presentation options and some contextual information for an elementary scene. In this way we condensed all information obtained from the strategic and the usage level into a sophisticated conceptual model.

In this section we explore further usages of web interaction types. The key observation is that a web interaction type does not need to be bound to an elementary scene, but could also be used as a conceptual means for a non-elementary scene, i.e., it could provide data and functionality as well as all other extensions for a complete scenario. There are several possible interpretations for these usages that go beyond interface abstraction:

1. If a web interaction type is associated with a complete scenario, it could serve as a formalisation of a navigation context emphasising the scene hierarchy. We will discuss this usage in detail in Section 8.2.2.
2. A web interaction type may also be associated with a session, which we may understand as a dynamically built scenario. In this case each of the web interaction objects would correspond to a session of a user from its start to its end. This generalisation of scenes by means of web interaction types will be discussed in Section 8.2.1.
3. The third usage of web information types on higher levels of abstraction concerns collaboration providing an abstraction from individual actors to group actors. For this we will exploit exchange frames for cooperation, communication and coordination in Section 8.2.3.

8.2.1 Web Interaction Types Associated with a Session

As sketched above the idea is to exploit web interaction types for non-elementary scenes in the storyboard to support the concept of a session. Informally, a session comprises all scenes visited by a user from entering the systems until leaving it. Exploiting the hierarchies of scenes, a session can be represented itself by a non-elementary scene. That is, entering a session refers to the creation of the corresponding web interaction object, which is linked to the elementary web interaction objects used for defining subscenes. Technically, we can model this by links from the web interaction types associated with the elementary scenes to the overarching session scene. Such links can be modelled as being hidden, i.e., they actually do not appear as part of the navigation structure. The session object may represent data that is carried around through the session. A typical example is the shopping cart in e-commerce systems. It is deleted by means of garbage collection, i.e., when no more web interaction objects exist that link to it.

Let $W = \{M_0, \dots, M_k\}$ be a set of web interaction types such that each M_i is associated with a scene Sc_i in the storyboard. Let us assume that this set is *navigation-closed* in the sense that each M_{i+1} can be reached from M_i either by a link defined by the content data type t_{M_i} or through an `Open`-statement in some operation in $O(M_i)$.

That is, it is possible for a user to start from a web interaction object (u_0, v_0) of type M_0 and successively create web interaction objects (u_i, v_i) of type M_i . In other words, the sequence $(u_0, v_0), \dots, (u_k, v_k)$ defines the prefix of a session of the user.

Definition 8.8. Let $W = \{M_0, \dots, M_k\}$ be a navigation-closed set of web interaction types. A *session type* for W is a web interaction type M such that each M_i is linked to M , i.e., the structure expression $\text{cont}(M_i)$ contains a pair $\ell_i : M$.

If M is a session type for a navigation-closed set W of web interaction types, then a *session* consists of

- a sequence $(u_0, v_0), \dots, (u_k, v_k)$ of web interaction objects resulting from the navigation through the scenes associated with W ,
- a sequence of web interaction objects $(u, v'_0), \dots, (u, v'_{\ell})$, all of type M , such that v'_{i+1} results from execution of an operation in $O(M)$ on (u, v'_i) , and
- an assignment $(u, v'_i) \mapsto (u_{f(i)}, v_{f(i)})$ with a monotone increasing function f .

Informally, in the scene associated with the web interaction object (u_i, v_i) a user may several times access the associated session object (u, v'_j) and modify its value. When progressing to the next scene associated with (u_{i+1}, v_{i+1}) , the session object remains the same, but can again be modified. In addition, the session object may be modified by operations in $O(M)$ itself.

Example 8.9. Let us simply look at the cart example. In this case the cart defines a session type that is linked by all other web interaction types of the system that contain an operation *add_to_cart*. The selection type for this operation on M_i may be a set type $\{t_i\}$, so the content type of the session type could be $\{(a_1 : t_1) \oplus \dots \oplus (a_k : t_k)\}$. It might also be simply $\{t\}$, if all t_i are the same and equal t . The body of the operation could simply access the linked session object and modify its value v by inserting the values in the selected set of values.

Furthermore, as there is a link from M_i to the session type M , a user may navigate to the session object (u, v) associated to the current web interaction object (u_i, v_i) of type M_i . In doing so, he or she could then modify v by means of some operations in $O(M)$. These operations can be the deletion of elements in v – this is possible, as v is a set value – or the common *proceed_to_checkout* operation associated with the session type.

8.2.2 Context Injection

Another usage of web interaction types for complex scenes is the support of the navigation context by exploiting the hierarchy of scenes in the storyboard. For this assume that S is a non-elementary scene in the storyboard that is defined by a scenario Sc . At its core Sc is defined by a directed, labelled graph, the vertices of which are again scenes. Therefore, let S_1, \dots, S_k denote the scenes of Sc and assume that each S_i is associated with a web interaction type M_i . The base case that all the scenes S_i are elementary has already been covered.

Definition 8.10. Let M_1, \dots, M_k be web interaction types that are in 1-1 correspondence to the scenes of a scenario Sc of the storyboard. A *context type* over Sc is a web interaction type M such that each M_i contains a link $\ell_i : M$ in its structure expression $cont(M_i)$.

Note that it is permitted that a web interaction type is linked to several context types.

Let us look back at the definition of context in Section 8.1. There we defined a context $C = \{t_1 : M_1 \rightarrow C_1, \dots, t_\ell : M_\ell \rightarrow C_\ell\}$ to comprise web interaction types M_i , from which we could reach the current web interaction type. More precisely, we could reach the current web interaction object from a web interaction object of type M_i . By means of the associated context C_i we could further navigate to other web interaction objects thereby following back a navigation path. On this path a user would collect condensed information by means of values of the data type t_i . That is, the context used in the contextual extension refers to web interaction types in the same scenario, whereas the context type defined above covers the whole scenario.

Consequently, when a user navigates through scenes in the scenario Sc , he or she may each time update information associated with a *context object*, i.e., a web interaction object of the context type M . This update may be done implicitly as part of the operations associated with the web interaction types M_i or M_i may even provide explicit operations affecting the context object.

Example 8.11. The content expression $cont(M)$ of a context type may associate with the name S of each scene in Sc a data type t_S comprising the combined information extracted and condensed from this scene in one or more visits. It may further provide links among those scenes that permit a step-by-step perusal through the previously visited scenes. In this way, accessing the context object a user would be enabled to backtrack through his or her navigation path with the advantage that only information is provided that is necessary for the user to keep him or her informed about the navigation history.

Alternatively, the content expression $cont(M)$ of a context type may contain simply an aggregation of all data of web interaction objects visited on the navigation path through Sc . In this way a simple look-up of the context object would allow the user to restore information about his or her past navigation.

In addition to the usages of context types in Example 8.11 any additional contextual information that is relevant for a user in scenario Sc may be associated with a context object.

8.2.3 Collaboration in Web Information Systems

The third usage of web interaction types for higher-level abstraction concerns collaboration among individual actors that form members of a group. For instance, a group may have tasks that are to be solved by collaboration

of several users, e.g., using group work in e-learning systems. Collaboration is understood as combining cooperation, communication and coordination. Therefore, the ground idea for capturing collaboration in WISs is to exploit a web interaction type for a complex group activity that links to web interaction types of individual actors. The cooperation, communication and coordination is managed by the overarching group type.

To formalise this idea consider again the scenes of a scenario Sc defining a scene S , each associated with a web interaction type. However, in this case assume that this set is partitioned into subsets associated each with a different actor. Formally, let $\mathcal{A} = \{A_1, \dots, A_k\}$ be a set of actors, and let $MT(A_i) = \{M_1^i, \dots, M_{n_i}^i\}$ be a set of web interaction types associated with the actor A_i such that $\bigcup_{i=1}^k MT(A_i)$ is the set of web interaction types associated with the scenes of the scenario Sc . As this is equivalent to an assignment of actors to the scenes in Sc , we call the pair $(\mathcal{A}, \{MT(A_i)\}_{i=1}^k)$ a *grouping arrangement* for S .

We can express the collaboration arrangements by means of an overarching web interaction type M associated with the scene S , which we will call a *collaboration type* for the grouping arrangement. Naturally, each M_i must contain a link $\ell_i : M$ to the collaboration type in $cont(M_i)$, but also M must provide links $\ell_i^{\text{prime}} : M_i$ in its content structure expression $cont(M)$. By means of these links each web interaction object of type M_i can identify its associated collaboration object of type M and vice versa.

Now M will take care of the cooperation, communication and coordination among the actors in \mathcal{A} and their web interaction objects.

For *communication* the exchange of messages between the agents (or more precisely between the associated web interaction objects) is maintained by the collaboration object. For this $cont(M)$ must contain incoming and outgoing message pools for each of the agents A_i . Sending of a message by A_i from a web interaction object (u_j^i, v_j^i) of type M_j^i requires an operation *send* in $O(M_j^i)$, which takes data from a selection type and additional input from the user, and changes the corresponding outgoing message pool for the associated collaboration object (u, v) of type M , which triggers an operation to change the content of the receiver web interaction object $(u_{j_r}^{i_r}, v_{j_r}^{i_r})$ of type $M_{j_r}^{i_r}$ associated with actor A_{i_r} , the receiver. This communication can be synchronous or asynchronous, so the initiating sender may have to wait for a response or not.

This base form for the communication between web interaction types associated with different actors may be enhanced by sophisticated handling on the side of the collaboration object. This may involve postponing messages, rejecting messages, waiting for conditions to be satisfied before deciding about the next action, etc. These are all part of the specification of the operations in $O(M)$ that will be triggered by the arrival of a message.

Regarding *cooperation* the communication object (u, v) contains all information about the status of the execution of the task, i.e., the progress with

respect to scenario Sc , by means of its links to the web interaction objects associated with the various actors of the grouping arrangement. By means of this the collaboration object may influence the scheduling of the action flow by re-assigning tasks to individual actors, which may again be triggered by conditions satisfied by the content v using operations defined in $O(M)$.

Finally, regarding *coordination* the collaboration object (u, v) permits the detection of inconsistencies among the web interaction objects of the actors in the grouping arrangement, which may trigger repair operations that are defined in $O(M)$.

Collaboration types may be associated with an actor, in which case decisions and the invocation of operations will be done by this actor. This actor may be elected by the group \mathcal{A} of actors as a leader, and the leader may change over time. It is also possible that collaboration types are hierarchically structured according to the scene hierarchy. Alternatively, the collaboration object may be associated with a passive actor who always reacts by means of triggering operations in case its own value v is updated by one of the actors in \mathcal{A} .

Example 8.12. The collaborative practical training (KoPra) system [796] has been developed by the German University Notebook Initiative. Its aim was to support adaptive and collaborative learning. The system is supporting communication and interaction of learners, organization and coordination of collaborating communities, free access to material depending on the progress, roles and portfolios of partners, and the stepwise development of training material. The system development has been forced by the change of the affiliation of the first and third authors from Cottbus to Kiel and the need of providing a practical training in database programming for attendees of the “Databases I” course in Cottbus. Learners act in a collaborative setting depending on their needs and the goals of the learning program.

The KoPra architecture separates the communication, the cooperation and the coordination in a layered way. It is a special case of collaboration type approach by means of web interaction types as described above. The KoPra application supports collaborative learning of learners. A learner

- may be a member of several learning groups,
- has a personal profile and a learning portfolio that consists mainly of learning tasks,
- can be the author of different kinds of documents, and
- uses different web interaction objects depending on his participation in learning groups.

Web interaction objects may additionally be delivered to working places that are assigned to learning groups in order to support their work. Principles of learning and teaching are multi-codality and multi-modality with reference to the learning context depending on the learning situation and the collaboration situation. The teachers and learners play active roles in the learning

process. The teacher acts as a coach, advisor, and moderator. Learners act in collaboration with other learners.

8.3 Bibliographical Remarks

Kleene algebras (KAs) and Kleene algebras with tests (KATs) have been, for instance, investigated in [449, 452, 881]. For both KAs and KATs interesting decidability and completeness results hold [451, 456]. If we ignore assignments, KATs can be used to model abstract programs. Doing this, it has been shown in [453] that KATs subsume propositional Hoare logic [321].

Context has been usually defined within the object sets of the database [66, 159].

The interaction engine approach has been generalized to a general theory of interactive information systems in [281].

Key Messages

Web Interaction Types can be extended

- to support different *measures* associated with the values in web interaction objects, and associated conversion operations;
- to support different *orderings* associated with bulk data in web interaction objects, and associated reordering operations;
- to support different *presentation options*, and associated operations to change ambience and presentation;
- to support *contexts* that permit to gain an overview of the navigation history.

The **Usage of Web Interaction Types** can be extended

- to support information about the current *session* associated with a web interaction object;
- to support condensed *contextual information* about the navigation path associated with a web interaction object;
- to support *collaboration* between different actors in a scene.



Screnography

The development of a WIS finally boils down to the writing or generation of web pages. So, whatever sophistication has been achieved through the method, a poor layout of the pages can easily destroy it. Nonetheless, astoundingly little effort has been put into WIS layout. It may be argued that layout is the realm of HCI techniques [69, 140, 379, 887], but then general HCI techniques are not coupled with specific development methods, and a lot of the HCI ideas have to be taken into consideration already during WIS design, even at a very high level of abstraction. For instance, the partitioning of pages and colour schemata are linked to the strategic decision on the desired ambience of the WIS as emphasized in [586, 581].

Graphical user interfaces are commonly considered to be a crucial component of web sites. They impact on the acceptability, usability, and reliability of the site and its compatibility with different environments, and thus also on the costs of site utilisation. Due to the complex requirements arising from the usability request user interface development is commonly considered to be more an art rather than an engineering discipline. Consequently, the support of high-quality web page layout through style sheets, inclusion of multimedia elements, navigation elements, texture, etc. has been investigated quite intensively, leading to a large number of publications on the subjects, e.g., [893].

As late consideration of graphical issues may result in inflexibility and cause problems for extension and change management, the aim of this chapter is to support the systematic and early involvement of layout and playout, for which we develop an approach to *screnography*, which adopts and generalises dramaturgy and scenography. Scenography has its roots in theatre, film and television, i.e., outside the web area, and contains the composition of action space, plot and dramaturgy. Dramaturgy controls the sequence of scenes and determines the composition of information. Our claim is to show that WIS layout also requires scenography and dramaturgy to facilitate the understanding and memorisation of content, and to support orientation within the WIS.

Web engineering as presented in the previous chapters is the application of systematic and quantifiable approaches (concepts, methods, techniques, tools) to cost-effective requirements analysis, design, implementation, testing, operation, and maintenance of high-quality web applications [390]. As content has to be presented to a highly heterogeneous audience of web users, content modelling and adaptation requires an integrated general concept for presentation.

Screenography aims to provide interfaces of high utility that can be used by any user depending on his or her intentions and tasks. Therefore, we base screenography on a characterisation of tasks, intentions, and specific characteristics of the users, which are provided by means of user profiles and portfolios as discussed intensively in Chapter 5. Intentions and tasks determine a user's expectations and interests, and specific characteristicitics influence the patience in dealing with the WIS. As WISs are to provide needed services, the users' real needs and life stories have to be specified. We do this on the basis of life cases that generalise, combine, extend and formalise business use cases. As users have different personalities, cognitive aspects are taken into consideration.

Screenography captures the *layout* and *playout* of WISs. Layout addresses the graphical design of pages in liaison with content to be delivered and functionality to be provided by the WIS. It supports users depending on their profile. Playout is based on the story specification, the task portfolio of users, and the contexts of the WIS and its users.

9.1 Development Prerequisites

The prerequisite of screenography is the analysis of a WIS on a high level of abstraction. Therefore, we have to consider the intentions associated with the WIS, and characterize expected users. This is captured by a storyboard that specifies who will use the system, in which way, for which tasks, and for which intentions.

Development Architecture

The Seeheim model for user interface management systems (UIMS) [287] separates the presentation from the application components. Consequently, the development steps regarding these components are separated, too.

The *application* component distinguishes levels for requirement prescription, specification and implementation. However, for the *presentation* component only the specification and implementation level are dealt with thoroughly at present. Nevertheless, the requirement level is also important for the presentation component. At this level we define the intention to be supported by the presentation and characterize the WIS users. For instance, we may define the ambience of a decoration to be colour-independent, because the perception and impression of colours depends on capabilities of users as well as cultural

and religious views. On the specification level these global definitions function as guiding conditions.

Facets of Intentions

The motivation of a user for WIS use is explicitly specified through four facets of intentions: *purpose* (aims or objectives), *intents* (targets or objects), *time* (design, end or occasion) and *representation* (atmosphere or metaphors). Roughly speaking, the first facet represents the ‘what’, the second the ‘how’ and the third the ‘when’ of an intention. Details of these facets were presented in [735].

User Models

User modelling is based on the specification of *user profiles* that address the characterisation of the users, and the specification of *user portfolios* that describe the users’ tasks and their involvement and collaboration on the basis of the mission of the WIS [730].

To characterize the users of a WIS we distinguish between *education*, *work* and *personality* profiles. The education profile contains properties users can obtain by education or training. Capabilities and application knowledge as a result of educational activities are also suitable for this profile. Properties will be assigned to the work profile, if they can be associated with task solving knowledge and skills in the application area, i.e., task expertise and experience as well as system experience. Another part of a work profile is the interaction profile of a user, which is determined by his frequency, intensity and style of utilization of the WIS. The personality profile characterises the general properties and preferences of a user. General properties are the status in the enterprise, community, etc., and the psychological and sensory properties like hearing, motoric control, information processing and anxiety.

A portfolio is determined by responsibilities and is based on a number of targets. Therefore, the actor portfolio (referring to *actors* as groups of users with similar behaviour) within an application is based on a set of tasks assigned to or intended by an actor and for which he or she has the authority and control, and a description of involvement within the task solution [730]. A *task* as a piece of work is characterized by a problem statement, initial and target states, collaboration and presupposed profiles, auxiliary conditions and means for task completion. Tasks may consist of subtasks. Moreover, the task execution model defines what, when, how, by whom and with which data a task can be accomplished. The result of executing a task should present the final state as well as the satisfaction of target conditions.

Life Cases

For task completion users need the right kind of data, at the right time, in the right granularity and format, unabridged and within the frame agreed

upon in advance. Moreover, users are bound by their ability to verbalise and digest data, and their habits, practices, and cultural environment. To avoid intellectual overburdening of users we observe real applications before the system development leading to *life cases* [732]. Life cases help closing the pragmatic gap between intentions and storyboarding. Syntax and semantics of life cases have already been well explored in [735].

Demands for Presentation Development

Intention, user and life case modelling are very important for screenography. For instance, for developing a satisfactory WIS atmosphere we have to consider the representation facet of the intention, which defines the ambience the presentation should have. It depends on the application area of the WIS and the preferences of the user. The main challenge is the heterogeneity of web clients. Therefore, we specify the atmosphere already on a high level of abstraction.

Life cases also depend on the defined intentions. By specifying a life-case-based story flow it is possible to categorize them and deduce presentation requirements. Moreover, the design preferences of users are very diverse, so we have to adapt the decoration to the expectations to get a suitable presentation, whereas default values or manual selection are of minor importance. Therefore, we characterize the users by profile and portfolio and deduce presentation preferences. It should be noted that too many changes impair the orientation of a user, so the challenge is to be at the same time adaptive and consistent with the user preferences.

9.2 Elements of Screenography

Screenography can be seen as the web analogue of scenography. It extends web application engineering by scenographic and dramaturgic aspects and intends to support the interaction between system and user. Screenography aims at an individualized, decorated playout in consideration of user profiles and portfolios, provider aims, context, equipment and the storyline progress. Heterogeneity of web clients and their different capabilities provide challenges to screenography.

9.2.1 Atmosphere

Defining the atmosphere is the first step of presentation development and a part of intention specification [586]. Due to its definition on a high level of abstraction the atmosphere is independent of equipment features. The reification through page layout and the sufficiency of the available equipment to play out a specific atmosphere will be checked on a lower level. The atmosphere can be

described by the ambience of the presentation, and thus it is not only determined by a colour schema. Ambience is also determined by other parameters such as shapes, material and illumination. Nevertheless, visual perception is always affected by the current mood and emotions of a viewer. According to [580, 581] we distinguish the following ambience-types:

- *powerful* in the sense of dramatic art and vitality,
- *romantic* in the sense of romance and passion,
- *elegant* in the sense of seriousness and dispassion,
- *refreshing* in the sense of ease and transparency,
- *balanced* in the sense of harmony and balance,
- *energetic* in the sense of phantasm and energy.

[Figure 9.1](#) presents some ambience examples taken from [580, 581]. It shows background facets of the same application, in this case varying only by colour.

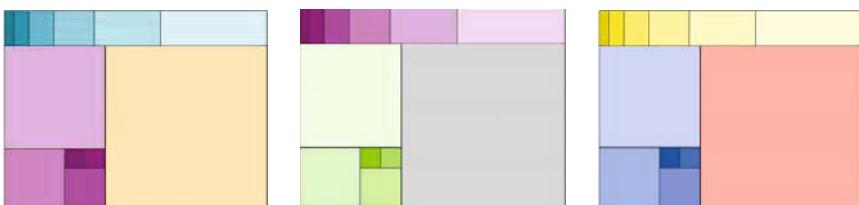


Fig. 9.1. Facets of the same application: refreshing, energetic, romantic ambience

9.2.2 Atmospheric Effect of Colour Schemes

The desired atmosphere of a WIS in [Figure 9.1](#) uses categories such as energetic, romantic, elegant, refreshing, harmonic and stimulating. All these categories refer to a desired sensation that the WIS presentation should convey to the users. The question is, which colour schemes can be identified to support these desired effects.

For each category we first need a ground colour chord, which is extended to a quality contrast. In particular, colours will be brightened uniformly. Based on these base decisions we develop an ordering of colours. The colour ordering in combination with the association to grid tiles determines the overall effect of the colour scheme. The colour ordering is based on formal and functional criteria:

- Formally, the order is determined by the placement of the colours in the colour circle. It is known that equidistant colours harmonise better.
- Functionally, the order of colours is determined by the subjective sensation and associations of a viewer. This is taken from psychological studies and centuries of experience in art.

An energetic or powerful atmosphere can be achieved using a three-colour chord with bright and high-croma colours such as a blueish purple with green and orange. The blueish purple colour with increasing brightness can be used for the visual flags. Orange with increasing brightness can be applied to the initial squares on the Fibonacci spiral, where green is reserved for the largest tile. However, a tile will always inherit the colour of its corresponding flag.

A romantic atmosphere can be achieved using a three-colour chord with pastel colours such as (light) blue, pink and yellow. Similarly to the energetic atmosphere yellow and blue in increasing brightness would be used for the flags and the initial tiles along the Fibonacci spiral, where pink would be reserved for the largest tile.

An elegant atmosphere can be achieved using a two-colour chord, e.g., red and green, in combination with grey. As before, grey would be reserved for the largest tile in the Fibonacci grid (e.g., Figures 9.3 and 9.4), while the other colours in increasing brightness would be used for the initial tiles and the flags, respectively.

A refreshing atmosphere can be achieved using a three-colour chord with light colours in a “temperature contrast”, i.e., the individual colours show opposite effects with respect to associated temperature. For instance, we might choose (light) purple, yellow-orange, and a blueish green such as cyan. The use on the Fibonacci is as before with yellow-orange for the largest tile.

A harmonic or balanced atmosphere can be achieved using a two-colour chord, e.g., dark blue and ochre, in combination with grey with grey being used for the largest tile. Similarly, a stimulating atmosphere can also be achieved using a two-colour chord, e.g., yellow-green with red-purple, in combination with grey. Figures 9.2 and 9.7 show sample pages for these atmospheric effects.

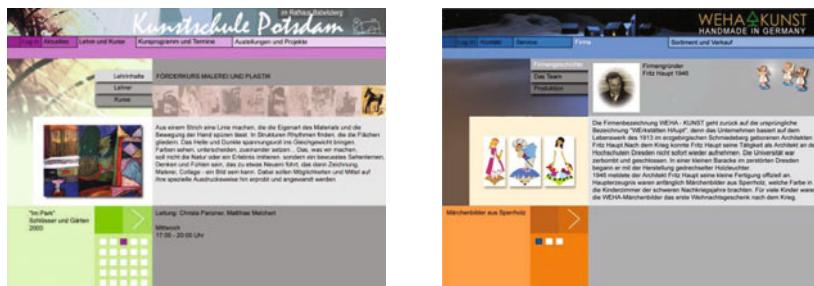


Fig. 9.2. Sample pages for stimulating and harmonic atmospheres

9.2.3 Layout Patterns

After defining the atmosphere, we have to specify layout patterns. Patterns are a powerful conceptual framework for building compelling, effective, and

easy-to-use websites [893]. A pattern consists of visual and functional building blocks. According to [586] the functional building blocks realise the access to the presented content and order these. The visual building blocks are important for perception and need to consider:

- the colouring with respect to functionality and aesthetics,
- the perspective perception of the whole screen, and
- the visual alignment and partitioning of the screen.

The colouring aspect includes the colour schema development on the basis of the specified atmosphere. Therefore, we have to consider the emotions a user usually associates with colours. The effect of colours can be warm, cool, cold, intensive, hot, light, dark, etc. A user's perception is also influenced by cultural aspects and age. According to Moritz the basis of a colour schema can be a colour chord consisting of n colours ($n \in \{2, 3, 4, 6\}$) that form a regular polygon in the CMY-based colour circle [580]. The colour chords can be complemented to a quality constraint by changing the saturation.

By using any kind of perspective it is possible to realize three dimensional decorations. The perspective aspect of a pattern is mainly determined by the colour schema. In particular, we have to consider the effect of the colours, because the objects coloured by warm colours appear closer than those in cold colours. Warm colours are assigned to light, while cold colours are assigned to shadow.

The visual alignment is based on a tiling of the screen as a two-dimensional surface. In general, we can divide the horizontal and vertical axes using grid points $x_{min} = x_0 < x_1 < \dots < x_m = x_{max}$ and $y_{min} = y_0 < y_1 < \dots < y_n = y_{max}$. A tile is defined by a rectangular region $[x_i, x_j] \times [y_k, y_\ell]$. Then we use a partition of the whole screen into tiles.

Example 9.1. A very common tiling is obtained by using just 4 horizontal grid points $x_0 < x_1 < x_2 < x_3$, and only 3 vertical grid points $y_0 < y_1 < y_2$. Then define four tiles

$$\begin{array}{ll} \text{up} = [x_0, x_3] \times [y_1, y_2] & \text{left} = [x_0, x_1] \times [y_0, y_1] \\ \text{middle} = [x_1, x_2] \times [y_0, y_1] & \text{right} = [x_2, x_3] \times [y_0, y_1] \end{array}$$

Usually, the “up” tile is used for some menu bar, the “left” tile for navigation links, the “middle” tile for major content and the “right” tile for side options.

9.2.4 Grid Geometry

Grids were adapted from (conventional) graphic design and used for organizing page layouts, e.g., newspapers, magazines and other documents [893]. The tiling described above is a very common but simple grid that only divides rows and columns, without any other restrictions. More sophisticatedly, the size of visual building blocks can follow a rhythmic structure that can be expressed

by a sequence of positive integers. Then an observer perceives larger tiles of a sequence as being more important, in particular, if the sequence shows a monotonic pattern.

Due to its overwhelming use in art and architecture over centuries we are particularly interested in the Fibonacci sequence, which is defined by the recurrence $f_{n+2} = f_n + f_{n+1}$ with the starting values $f_1 = f_2 = 1$. This gives the well-known sequence 1,1,2,3,5,8,13,... It also gives rise to the *number of golden section* [586], which played an important role in art and architecture and appears naturally in nature.

[Figure 9.3](#) illustrates the visual flags model as a simple use of Fibonacci sequence. In fact it dates back to Leonardo da Vinci and orders sections according to some functional criteria. In this example the Fibonacci numbers (multiplied with a scaling constant) are used as horizontal grid points.



Fig. 9.3. Rhythmic structure of visual building blocks (grid tiles)

Another use of the Fibonacci sequence is to place squares with increasing side length f_i along a spiral as shown in the lower part of [Figure 9.4](#). In doing so we obtain a very nice tiling of the screen (*golden square*), in which the proportions of the square tiles are determined by the Fibonacci sequence. By combining with the visual flags we realize the so-called *Fibonacci grid model* [586].

If each tile is associated with a colour of a well-chosen colour schema, this enables a desired atmospheric effect as specified in the strategic WIS model [586]. The thesis [580] shows this combination of the Fibonacci grid with various colour schemata in different web application projects.

9.3 Cognitive Aspects of Screenography for Layout

Screenography bases screen and particularly WIS layout on cognitive psychology. The layout of WIS pages contains *functional elements* such as icons for navigation and functions, and *visual elements* for presentation of texts based on script and colour and of pictures and structures in different displays. The designed screen is regarded as a structured composition of different elements and is defined as screen layout. Screenography uses three kinds of principles

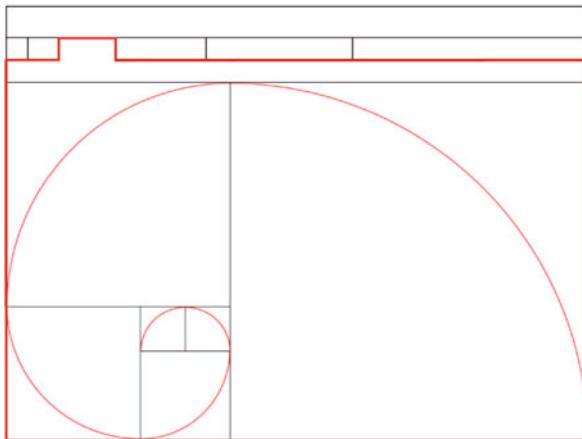


Fig. 9.4. The Fibonacci grid model

taken from cognitive psychology: principles of *visual communication*, principles of *visual cognition* and principles of *visual design*.

9.3.1 Principles of Visual Communication

A clear and well-defined design of a screen layout helps to grasp and to understand the content and enables selecting and accessing the information. It is a precondition for the successful communication. Visual communication is based on the exchange of visual codes with special meanings. Sender and recipient agree on the meaning of the communication utterances which are typically expressions in a visual language adopted and understood by both partners.

A sophisticated specification of visual communication is one precondition for interaction support. It consists of three components:

- *Vision* supports users depending on their physical and physiological properties. Users are used and are limited to certain colouring schemata, to different presentation styles and to different reception styles.
- *Cognition* is based on the physiological and psychological abilities and skills of users. We must take into consideration the approach users take while reading content and using functionality.
- *Processing and memorising characteristics* are based on the psychological ability to read, integrate, and reason about content provided by a page, and to memorize main parts of the content – these vary a lot among users.

Visual communication starts with separating the visual entities on the screen into elementary layout elements. This separation is an analytical pro-

cess. Visual elements are compared, processed and memorised after recognising them. The scenario for visual communication can be used for developing the layout, i.e., we reverse the order of visual communication. First, visual elements that should be memorised are developed and integrated with the content and functionality necessary for their recognition. Next, these elements are integrated into a draft of the layout. Finally, adornments, presentations, and placements are added.

The process of transferring experienced vision into analytical vision of visual information is based on specific visual features:

- *Contrast* allows to recognise visual information as such. The *format* is crucial for the function and meaning of visual information in a visual context.
- *Visual analogies* support recognition of targeted associations and enhance the meaning of visual information.
- The *picture dramaturgy* controls the selective observation of visual information and governs the recognition of their meaning in the visual context.
- The *reading direction* determines the order in which visual information is noticed in the visual hierarchy.
- *Visual closeness* of presentation is based on structural aspects and affects the context of visual information.
- *Symmetric presentation* of visual information refers to order and is better memorisable than asymmetric and disordered visualisation design.
- Space and movement are features of static visualisation. They are time-dependent. Movement is associated with the structure and the reading direction.

We explicitly use these visual features for development of visual elements as well as for their composition.

9.3.2 Principles of Visual Cognition

Principles of visual cognition and visual communication refer to *ordering*, *effect delivery*, and *visualisation*. Ordering is based on an arrangement according to the reading direction and on design according to foreground and background relation. The effect uses background for formation of thematic and optical frame, and schemes of colours and structures. Visualisation is influenced by visual design features such as colour, contrast, composition, overlapping, and cuts.

We can use a number of principles of visual cognition in screenography. Users are limited by their time, attention, scope, and task portfolio. These limitations must be taken into consideration for layout and playout development. We distinguish four principles that should be preserved:

- The principle of *organisation* requires that screens must be based on a simple, clear and consistent structure. The simplicity, clearness and consistency depends on the cultural background of users.

- The principle of *economy* requires that users should spend minimal effort for recognition and reasoning on visual elements. This principle is supported by minimisation of the set of tools. Users increase their effectiveness of recognition and reasoning.
- The principle of *communication* considers the abilities and the skills of users. This principle is well known from didactics, where content and functionality are adapted to the capacity of users.
- Screen design *standards* allow the user to reuse their previous recognition and reasoning results. These standards define the organisation and design of visual elements.

9.3.3 Principles of Visual Design

We use a number of principles of visual design in screenography:

- The *optical vicinity* principle requires that elements are arranged close to each other, if they are related.
- The *similarity* principle is based on human cognition according to which elements of similar shape are identified as a visual unit.
- *Closeness* is based on the human visualisation according to which objects need to have a closed shape. Otherwise, missing elements are added.
- The *symmetry* principle uses units that consist of symmetrically assembled elements. Symmetric and asymmetric structures are visualised in the foreground, whereas all other asymmetric elements are taken to the background.
- The *conciseness* principle uses visualisation with simplified and consistent organisation of visual screen elements.
- The *reading direction* principle optically composes static elements (pictures, texts, icons) by the given reading direction. This leads to a shift of the optical focus. We distinguish between geometrical and optical centre. For instance, the thickness of horizontal lines is optically enhanced. Vertical lines seem to be thinner.

These principles help to organise the elements within a page in a way that corresponds to human perception. Elements are always recognised within their context. Their value differs

- based on their syntax, i.e., the formal and aesthetic value,
- based on semantics, i.e., content and objective value, and
- based on pragmatics, i.e., ethic and applicability value.

Visual elements such as icons are associated by the content and functionality they display. They have their own visual attributes and meaning. Their composition, however, extends the interpretation and meaning, thus creating additional information for the user.

The human eye tends to simplify structures and reduces them to familiar things during recognition. Therefore, we target at a visual order based on harmony and balance. In addition, we may use familiar and well-known elements.

9.4 Application of Screenography: Case Study

In this chapter we demonstrate the potential of the screenography approach, using a B2C example. This example indicates how we achieve an adaptive playout, considering the intention of the WIS, user profile and portfolio as well as existing life cases. In general this B2C-WIS offers company and product information can described by following information pattern (who-what-to_whom-what_activity):

$$(company)^{information_on_products} 2(visitor, purchaser)^{inform, purchase}$$

In our example acts a small business, producing hand-made collections of wood-figures. The aim of company is to distribute the collections via the web. Considering the business philosophy, the presentation style has to be traditional and folksy. Wood as basic material of all products of the company should be considered too. Because of these requirements we select the ambience type ‘harmonic’.

Before defining the ‘harmonic’-colours we have to characterise the user. That’s necessary because the perception and impression of colours depends on capabilities of a user and e.g., cultural as well as religious views. Moreover the age of a user plays an important role. In our example the user is a European man in his thirties, without special colour preferences. The user prefers a conservative design and an adequate composition. In conjunction with the intentions we deduce a suitable colour chord illustrated in [Figure 9.5](#).



Fig. 9.5. Colour chord – ambience type ‘harmonic / balanced’

Subsequent to the atmosphere determining, we have to choose a pattern. Patterns consist of three parts: visual alignment, colourizing and perspective perception (Section 9.2.3). Visual alignment is influenced by life cases, perspective requirements and the user profile as well as main properties and targets. For instance the content size and the navigation structure and depth influence the tiling. The example user acts deliberate and target oriented. He has no handicaps but does not prefer very small interaction elements. We

consider lifecases to achieve the best possible orientation the user should have during the acting progress. Thus it's possible to choose the Fibonacci grid for representation, illustrated in [Figure 9.4](#).

The next step is the combining of colour chord and pattern. In which way the colours will assigned to the pattern mainly depends on the chosen ambience type. The coloured pattern is illustrated in [Figure 9.6](#).

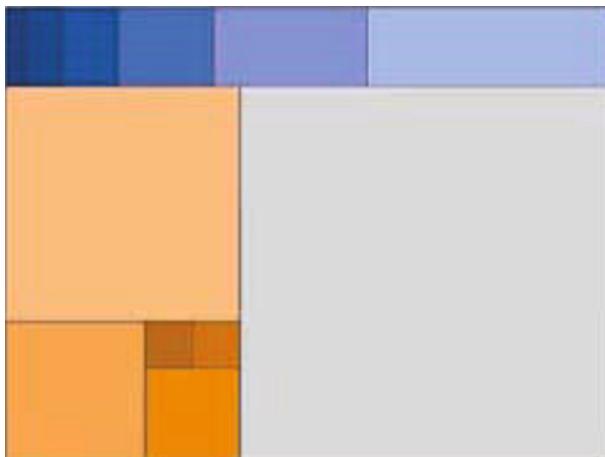


Fig. 9.6. Fibonacci grid model with ambience type 'harmonic / balanced'

The last step integrates the content. The representation of content considers user and provider preferences. For instance an adequate picture-size to present details of the hand-made products. User portfolio and life cases influence the story flow. Furthermore, depending on tasks they have influence in content representation, e.g., the type of progress. The result of the development process is illustrated in [Figure 9.7](#).

9.5 Screenography Guidelines and Frames

Screenography is a novel approach to layout and playout of Web Information Systems. It transfers the accumulated knowledge in scenography and dramaturgy from its origins in theatre, film and television to the web area thereby exploiting results from creative arts, cognitive psychology, and stage scenography. The rationale behind our work is that layout and playout are not merely activities that start when the major system design has been completed, but must be treated as integral part of WIS development from the very beginning; a poor layout can nullify all sophisticated modelling work, because the layout is the ultimate carrier of information in a WIS.



Fig. 9.7. Sample page for ‘harmonic / balanced’ atmosphere

Screenography is tightly intertwined with storyboarding, the method for WIS usage modelling on a high level of abstraction [727]. It contributes to page partitioning and colour scheme definition in a holistic way. Furthermore, it adheres to known principles of visual perception, communication, cognition and design that address important features such as contrast, rhythm, and perspective.

Screenography is an attempt to turn WIS layout and playout from an art into a craft, i.e., it aims at enabling WIS designers to engineer systems that by virtue of their presentation are conceived as exceeding the expectations of their users. The work presented in this paper is a first step in this direction, yet still much more has to be done to extract knowledge from arts and bring it into a form that can be used by WIS engineers.

We observe three generations of approaches to web page design and development:

- First generation web page design has been developing functions for graphics, general principles for page layout, and a number of tools supporting easy development of web pages. The content and the functionality dimension of website specification was integrated into design mainly through programming.
- Second generation web page design used already generating facilities to generate web pages layout by applying grids for generation of suites of websites. This generation is neatly supported by XML standards such as XSL

and a number of third-level standards. The content and functionality dimension of websites is already automatically integrated. Some approaches used website specification languages such as SiteLang and enabled the integration of storyboards and figures and active figures where the latter are based on profiles and portfolio.

- Third generation web page design is introduced and discussed in this paper. Within this generation automatic generation of web page grids is based on web page pattern and incorporates two of the six dimensions necessary for website specification: intention and context.

For the *third generation web page design* we develop a high-level description based on web page pattern, by mapping web page pattern to web page grids, and by generating the web page specification to *web page suites*. The method has been successfully applied in our new website projects. Our approach is currently used for extending the SiteLang editor [858]. Currently, the editor allows to generate the entire website based on the specification. The first implementation of our approach has shown that the editor can easily be extended by web page pattern and web page grids.

9.5.1 Web Page Pattern

Patterns emerge as abstractions of *web page design frameworks*, which we adopt from object-oriented systems development [161, 263, 312]. A *framework* is a reusable, “semi-complete” application that can be specialised to produce custom applications [379]. In contrast to earlier reuse techniques that were based on class libraries, frameworks are targeted for particular business units such as data processing or cellular communications and application domains such as user interfaces or real-time avionics. In other words, a framework is a reusable design expressed as a set of abstract classes and the way their instances collaborate.

Web page patterns are a generalisation and detailisation of the proposal [600] where the authors claimed that intentions of an application can be modelled through tasks and targets of accomplishing tasks. Following this claim we discover that the second, third, fourth, fifth and sixth dimensions of websites influence the layout of singleton web pages and the playout of suites of web pages. This association is displayed in [Figure 9.8](#).

Web page pattern describes the main principles that rule development of singleton web pages and depend on the remaining dimensions. For instance, community websites can support communities of transaction, forums, discussion groups, communities of interest (associations, focus groups), or communities of practice (following a process, working together). Page patterns are dependent on the story space, on the content and functionality dimension. They are specified by utilisation space, utilisation portfolio, and principles:

- The *characterisation of the utilisation space* is based on the kind of content, the kind of functionality supporting work, e.g., navigation facilities and

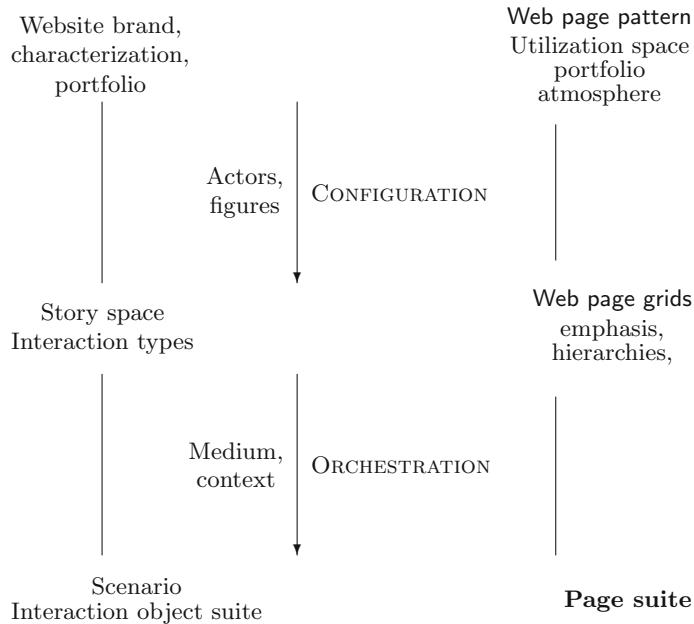


Fig. 9.8. The association of interaction types, the story space, page patterns and page grids

order principles, life cycle models for utilisation, principles of layout and playout of web pages and instruments used such as metaphors, and strains or the tone of the web site.

The utilisation space depends on the implementation of principles of visual communication (dialog recognition capacity of users, organization, economy of instruments), of principles of visual recognition (orderings, effects, organization), and of principles of visual shapes (optical closeness, similarity, unity, symmetry, pregnancy, recognizability). The utilisation space is also characterized by its navigational facilities (feedback, mental models for navigation, metaphors).

- The *utilisation portfolio* is based on the kind of tasks to be performed through the site, the story space of the website, e.g., the dramaturgy, and the intended audience of the website.
- General *principles* of the page design applicable to the entire website can be derived from the characterization of the brand of the website. Classification of websites based on the `who-what-to-whom-for-which-action` frame [727] (e.g., Business^{Product}2Customer^{byu} (or B2C), School^{Lecture}2Student^{learn}) can be used to derive “Gestalt” and visibility of a site and of pages.

The *atmosphere and the intention* of the website is ruled by categories such as *energetic and vital* (providing a flavor of progression and innovation),

harmonic and well-balanced (supported by soft coloring and design), or *inspiring and magical* (giving a sense of energy, vitality and pleasure).

Elements of page layout are surfaces, text, pictures and animations. Their integration is governed by the pattern requirements:

organization calm and close attention; lightness and weightiness;
harmony within a site and within pages, introduction of polarity to pages;
life and development within website and within pages.

The pages displayed in [Figure 9.9](#) provide information on the Potsdam Institute of Art, provide material for learners and should attract collaborators. So, the page should be harmonic, inspiring and magical at the same time. Furthermore, the pages must give an orientation with side pathes (left upper corner).

The utilisation portfolio of websites rules sequential stepwise visits of web pages. Web pages within a website do not follow the development of utilisation but use the same frame and appearance within the entire site. So, they are static and mainly linear. In most cases, however, websites have a story space that is neither linear nor static in its utilisation and recognition. We thus distinguish a number of *lifespan patterns*:

Evolution lifespan pattern records the *stages* of the life of things and/or their corresponding objects and are closely related to *workflows*.

Circulation lifespan pattern displays the phases in the lifespan of things.

Incremental lifespan pattern allows to record the development and specifically the enhancement of objects, their aging and their own lifespan.

Loop lifespan pattern supports nicely chaining and scaling to different perspectives of objects which seem to rotate in the workflow.

Network lifespan pattern allows the flexible treatment of objects during their evolution, supports passing objects in a variety of evolution paths and enables multi-object collaboration.

These lifespan patterns are to be supported by development grids such as the grids used in [Figures 9.9](#) and [9.10](#).

9.5.2 Web Page Grids

Page design, e.g., [592], is an essential element of scenography and of applied art. Current web page grids are based on equidistant, regular 2D placement. A typical grid is the three-column information placement with additional headers for navigation and additional navigation at the left side. If the amount of information becomes too large then mouse-directed roll down facilities are added to the page. Some web page also use 3D placement. These grids are appropriate as long as the amount of information to be displayed is rather small and there is no inner association among the content.



Fig. 9.9. Alternatives for the web site of the Potsdam Institute of Art [586]

Grids are specialized containers [823]. They have a number of parameters which are instantiated depending on the content, functionality and layout of site. Additionally, metaphors are used for the layout of singleton pages and for the playout of associated suites of websites.

Web page grids provide an orientation system in the utilisation space. It is based on unambiguous structuring, direct association and understandable context of content that is placed on a page.

Elements of grids besides the container specification are

- *form* and *coloring schema* as a unit of composition, including dimensions, artificial or natural origins, construction based on simple geometry or complex forms, and including properties of colorings,
- *contrast* and *rhythm* based on brightness, temperature, saturatedness, colourfulness, amount, form kind, size, dimension, Gestalt, structure, and place,
- structure and composition for content and functionality using specific visual structures (optics, layout, coding of information).

Building blocks based on containers may be classified into visual elements for layout and functional elements for playout.

- *Visual elements for layout* support optical organization and colouring schemes.
- *Functional elements for playout* through suites of web pages support usage, e.g., navigation.

The communication models for page grids combine functions and surface models:

functionality display, coding and hierarchy that does not suddenly change and supports surveying and ordering,
colouring and texture with differentiation through elements,
unity of movement and time for ease and unity of usage,
citations and analogies supporting recognition,
aesthetics for obtaining a unity, oneness of the pages, and
navigation and unity of navigational elements.

Patterns and grids may be tightly associated by generation functions:

$$\begin{aligned} \text{GenerateGrid} : & \{ \text{Pattern} \} \times \{ \text{ContentKind} \} \times \{ \text{EnvironmentKind} \} \\ & \times \{ \text{DialogStepKind} \} \times \{ \text{ActorKind} \} \mapsto \{ \text{Grid} \}. \end{aligned}$$

We assume that generation functions depend on parameters of the utilisation space, utilisation portfolio, and general principles. These parameters result in lifespan pattern. Kinds of content, context or dialogue steps characterize requirements such as content size and functionality, environment capacity, page restrictions, and barriers-freedom. The last three kinds are summarized as context kind [399].

Configuration of 2D Grids

We may distinguish a number of placement grids:

Linear grids are based on equidistant constant separation and constant functionality display.

Flexible grids use flexible separation and are preferred whenever change and time are going to be displayed.

The grids in [Figure 9.10](#) show how incremental lifespan patterns can be supported by grids based on a Fibonacci separation of the space. We added for orientation envelopes that allow to survey the stage of work.

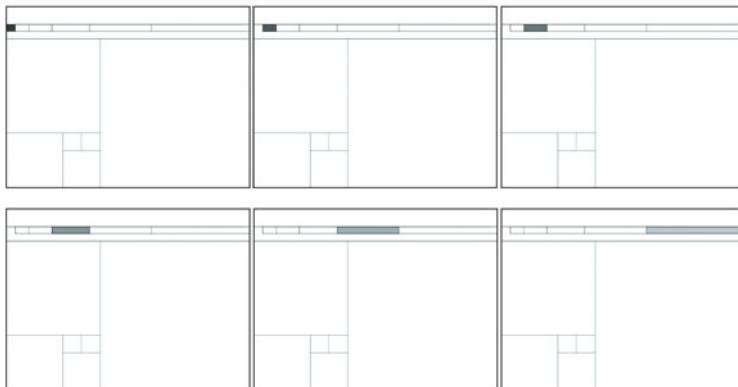


Fig. 9.10. Fibonacci separated grid set for lifespan-oriented websites with header envelopes

Orchestration through Context Injection

The concrete layout of web pages and the playout of web page suites depends on the content, the functionality, and the context. For web page generation through our tools we use the following three steps:

Enhancement for visibility by deriving and instantiating the coloring schema with values for unity, contrast, brightness, size, direction, quantity;

Enhancement for improvement of recognition by instantiation of form, color, movement, texture, space, and proportion parameters;

Enhancement for mental integration by adapting unity and level of detail, composition, perspective, and rhythm.

9.6 Bibliographical Remarks

The main sources for the chapter are the PhD theses [580, 620] (and research papers such as [579, 583, 584, 585, 586, 621, 618, 619, 675]) and the forthcoming book [582]. The screenography as important part of Web Application Engineering considers scenographic and dramaturgic aspects. The dramaturgy is a part of scenography and therefore it has its roots in the classical drama too. It controls the sequence of scenes and determines the composition of information.

Screenography aims at providing interfaces of high utility that can be used by any user depending on his or her intentions and tasks. Therefore, we base screenography on a characterisation of tasks, intentions, and specific characteristics of the users, which are provided by means of user profiles and portfolios. Screenography is based on principles of visuality such as visual communication, visual cognition, and visual design [580, 842]. It uses design patterns introduced in [131, 206, 217, 245, 277, 489, 592, 781, 880].

The screenography approach inherits also approaches to screenplay, e.g., [379, 887], communication theory and linguistics, e.g., [164, 309], and collage techniques [229, 561, 579]. Grid design is one of our features. Other sources to grid design are [591].

The design and reification of WIS cannot be left to “HTML-hackers” [491], but has become subject to various development methods such as OOHDM [762], WebML [106, 144, 560], WIS co-design [586, 709, 727, 739], HERA [334], contextual information systems (CIS) [818, 819], WSDM [183] and others, and also UML has been adapted to support WISs [158, 518].

These methods differ in many respects, and we do not intend to discuss these differences here. However, whatever method is used, it finally boils down to the writing or generation of web pages. So, whatever sophistication has been achieved through the method, a poor layout of the pages can easily destroy it. Nonetheless, astoundingly little effort has been put into WIS layout. It may be argued that layout is the realm of HCI techniques [69, 140, 379, 887], but then general HCI techniques are not coupled with specific development methods, and a lot of the HCI ideas have to be taken into consideration already during WIS design, even at a very high level of abstraction. For instance, the partitioning of pages and colour schemata are linked to the strategic decision on the desired ambience of the WIS as emphasized in [586].

Screen design follows principles of ergonomics [273, 301, 315, 792], typography [877], colouring [150, 278, 811], design approaches [623, 782], and psychology [140, 379]. Design itself is typically based on holistic principles that apply to the entire website, e.g., [179, 249, 804, 893]. Ingredients of screen design are the introduced principles of visualisation, see also [352, 423, 436, 561].

The simplest approach to screen design is the pattern approach [19, 17, 88, 219, 263, 522, 641, 911] that can be used to define stereotypes [502] which are refined to pattern and later to templates [122].

Literature knows many specification languages for interface development. UML is one choice [26, 158]. An advanced approach is the HERA language [249, 334].

User interfaces must be adaptable to the user, the current environment, to the infrastructure, and to the story. Adaptation requires still more research [656].

Key Messages

Screenography

- lifts the design of *layout* and *playout* of a web information system to an activity that is tightly coupled with the conceptual WIS model;
- combines the partitioning of the screen by *grids* with the atmospheric effects of *colour schemes* and the requirements arising from *life cases* and *user models*;
- exploits knowledge about *cognitive aspects* of visual communication and design;
- permits the coupling of static *layout elements* for the realisation of web interaction types with dynamic *functionality playout* for the realisation of the user actions.



Adaptation of Presentation to Culture

G. Hofstede defines the term culture as “a collective phenomenon, which is shared with people who live or lived within the same social environment, which is where it was learned; culture consists of the unwritten rules of the social game; it is the collective programming of the mind that separates the member of one group or category of people from others.” [327]

It is well known that the cultural background of a user determines whether or not he or she will find a particular presentation interesting, attractive or repelling. However, it can be observed that for most web information systems the user background, context, profile and portfolio are not properly taken into account. Therefore, this chapter is dedicated to the investigation of the following questions:

- Can information systems be partially adapted to specific user profiles and portfolio?
- Can we use user stereotypes for such adaptation?
- How can users be classified according to their culture and their specific behaviour?
- How can a WIS be adapted to the context of a user?
- Can context also be classified and systematically specified?
- Is it possible to use this classification for system adaptation?
- Is database technology capable to support WIS adaptation?

Hofstede’s notion of culture is a very general one. However, it permits to show how to automatically adapt the presentation system of a WIS to the specific culture of users. The support of cultural diversity has been investigated thoroughly in [354, 360, 361]. Here we follow these lines of research and introduce a general mechanism that allows a designer to adapt a WIS to cultural aspects.

Web information system development challenges classical technology and software engineering. There are heaps of open problems, especially problems associated with development of *SMART* systems (*Simple* in any step of usage

that a user might request; *Motivational* for any user independent of his or her working habit; *Attainable* for the goals a user has in mind; *Rewarding* as their investigation seems to be worthwhile and right in time, and matches efforts and needs; *Time-efficient* within the limits and expectations of users).

10.1 Understanding Cultural Differences

10.1.1 The Layered Structure of Culture

Hofstede's definition [327] refers to national cultures, which he sees as a layered (pyramid) structure including inherited and learnt properties that are finally adapted in the level of personality within the individual behaviour.

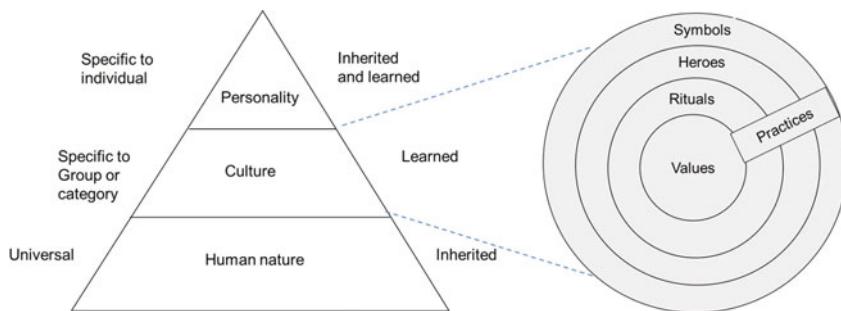


Fig. 10.1. The layered structure of culture [327]

The lowest level of the pyramid (left part of Figure 10.1) concerns human nature. It is common for all human beings, inherited by birth, universal and independent of group membership, and represents the “operating system” of the “programmable” human mind. The second level concerns culture, which is specific to a group emphasising the “collective part of the program”. This is illustrated in more detail by the onion model in the right part of the figure. The top level of the pyramid concerns personality, which is specific to an individual. It includes both inherited and learnt properties, which represent “personal mental programs”.

The onion model (right part of Figure 10.1) illustrates in more detail the elements of culture. Values are the core of a culture and characterise the preferred states. Rituals are collective activities related to a group of people. These rituals – like saying hello, ways of handshaking, etc. – indicate the togetherness of the members of the group and define its commonly accepted social norms. Heroes represent highly-prized persons as role models for the behaviour in a culture. The outermost layer concerns symbols, i.e., words, gestures and objects that are common for those that share the culture. Practices are visible elements of the behavioural patterns of individuals. They are

manifestations of the components having their source in different layers of the onion in the form of behavioural patterns, which are possible to interpret by knowing the roots of the elements. This forms also the base for the stereotype-based analysis of cultural differences.

Values form the base of a national culture with culture-dependent practices as manifestations. The “collective program of the mind” in this definition indicates learning capacity of the individuals, in which the topics learnt are organized according to the inherited part of the “program”. A human being is learning-intensive and adaptable, which means that ultimately his or her culture consists of elements recorded from the lifeline experiences that have their source in regional (in addition to national) culture, organizational cultures (e.g., based on the work experience), professional culture (based on education), etc. The learning aspect is also seen in Reinecke’s studies [670], where the term “*user’s extended national culture*” is defined as storage of the influence of foreign cultures to the national one, when a person is living in a foreign culture. These experiences change and enrich the original one. In addition, the national culture is also adapting under external influences. While the core remains the same, outer layers are adaptive.

10.1.2 Kinds of Culture

As stated above the components of culture are partially inherited and partially learnt. Culture consists of basic elements (as illustrated in [Figure 10.1](#)) and a stock of life experiences originating from several different aspects (see [Figure 10.2](#)).

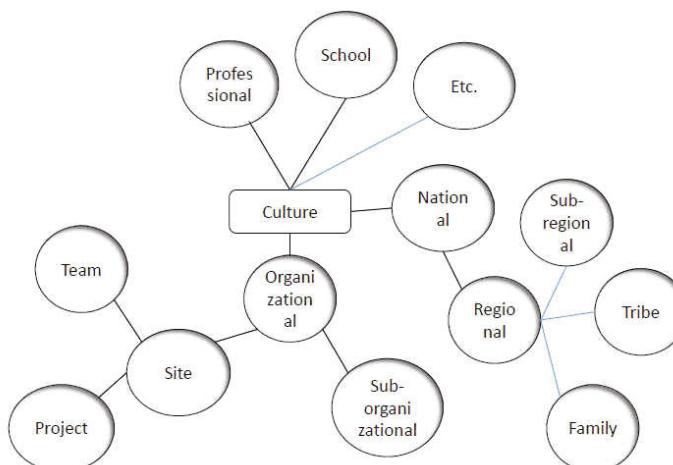


Fig. 10.2. Multidimensional aspects of culture

National culture is the most dominant constituent for the behaviour of individuals [360]: language, educational tradition, religion, beliefs, attitudes, and social context are important parts of a cultural identity. In addition, different levels of organisation-related cultures (organization, work, team, project, site), professional culture, regional culture, etc. represent collective similarities of groups of people with the same national heritage, and provide means for variations inside a national culture. These may cover the similarities in behaviour, interaction, decision-making, organizational structure, and goals. Besides themselves people represent views of different roles in parallel to an individual person's point of view.

Cultural aspects like language, educational tradition, religion, beliefs, attitudes, and social context are important sources of cultural dimensions, too. It is also worth recognising that national cultures have variations called *regional cultures*, which are variations of the collective part of the “program” of the mind.

Organisational (work) culture includes habits adopted by an organisation. It covers the similarities in behaviour, interaction, decision-making, organisational structure, and goals. People who have adopted the same organisational culture are able to communicate and transfer knowledge better than people from different work cultures. Analogous to national cultures also organisational cultures may have variations: *site culture* covers the aspects typical to one operative site of an organisation, even projects and teams may develop their own variations. *Professional culture* has its roots in education and adopted practices that are typical for certain professions. *Project culture* and *team culture* are cross-sections of organisational and professional culture.

The basic element of *individual culture* comes from birth having national elements as dominating factors. These are “tuned” by regional elements and further elements originating from smaller geographical areas. Other important aspects are related to organizational context having several branches and levels (sub-organizations, site, project team, etc.) and professional context (education, work experience, etc.). It is also important to notice that in addition to being summarised elements the cultural base is further influenced by exclusive roles. This is called *situational culture* emphasising that the same individual can appear in different situations as a representative of different roles, each time representing different (sub-)cultures.

10.2 Cultural Stereotypes

The most common approach in analysing differences between national cultures is based on cultural stereotypes. Stereotypes should satisfy at least five properties: (1) they must be accurate; (2) the quality of the stereotype allows that it is used consciously; (3) they should be descriptive, not evaluative; (4) they should be flexible so that they can be modified from time to time; (5)

they can be used as a first “best guess”. They provide generalised knowledge of factors related to national cultures.

Most commonly used and referred frameworks have been published by Geert Hofstede [323, 326, 327, 325] and Richard Lewis [502]. Additional approaches supporting our goals are provided by work on information system usage as handled by Reinecke and Bernstein in several papers [668, 670].

10.2.1 The Hofstede Model of Cultures

The model of Hofstede is based on the analysis of six cultural dimensions:

- *Power Distance* (PDI): the extent to which power differences are accepted;
- *Individualism / Collectivism* (IDV): the extent to which a society emphasises the individual or the group;
- *Masculinity / Femininity* (MAS): refers to the general values in the society distinguishing hard and soft values;
- *Uncertainty avoidance* (UAI): refers to the extent that individuals in a culture are comfortable (or uncomfortable) with unstructured situations;
- *Long-term / Short term orientation* (LTO): refers to the extent to which the delayed gratification of material, social, and emotional needs are accepted;
- *Indulgence / Restraint* (IVR): acceptance of enjoying life and having fun vs. controlling the life by strict social norms.

Every national culture has an index value for each dimension, which indicates its tendency in this dimension. Comparison of index values between cultures provides means for the need of adaptation in collaboration context. Hofstede’s resource pages [324] provide a tool for comparison of selected national cultures. The idea is to provide easy to notice differences between selected cultures. Hofstede’s work is applied by several studies to indicate preferences of people in certain contexts; e.g., Reinecke’s work applies it in user interface structure and look-and-feel of it. The culture model of Trompenaars [868] is a variation of Hofstede’s model.

10.2.2 The Lewis Model of Cultures

Richard Lewis is a business consultant who has long experience (since the 1950s) in training organisations to interact successfully in a global context. He has synthesized his experience in the triangle model that classifies national cultures in three basic categories [502, 504].

The model recognises three *basic stereotypes of cultures*. *Linear-active culture* is task-oriented and value is given to technical competence and facts. They are cool, factual and decisive planners. *Multi-active culture* is extrovert and human force is seen as an inspirational factor. They are warm, emotional, loquacious and impulsive. *Reactive culture* is people-oriented and dominated

by knowledge, patience and silent control. They are courteous, amiable, accommodating, compromisers and good listeners. National cultures locate on the sides of a triangle using the three basic stereotypes as angles. Figure 10.3 illustrates a slightly modified version of Lewis' model.

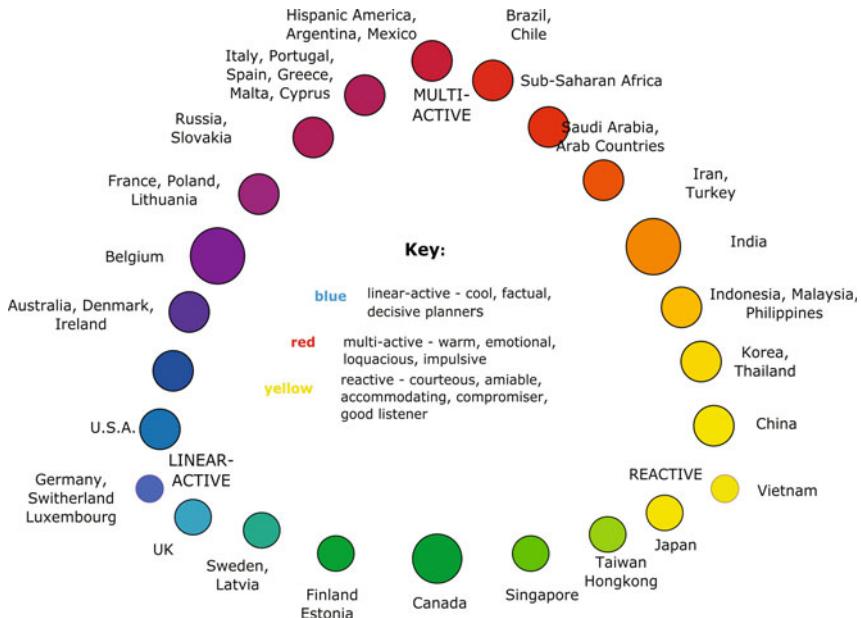


Fig. 10.3. National cultures in Lewis' model (revised)

Lewis has collected common traits of the three basic categories [502, pp. 33f.]. Those characteristics that are most relevant for our purposes are picked up in Table 10.1.

The Lewis model gives a detailed analysis of every national culture from four factors of point of view: *general facts* (geography, history, politics and economy), *culture* (general classification, values, cultural black holes, concept of time, concept of space, self-image), *communication* (communication pattern, body language, listening habits, audience expectation) and *interaction* (concept of status, gender issues, leadership, management, motivation factors, meetings, negotiating, contracts and commitments, manners and taboos, how to empathise). The country description available in internet resources (see e.g., [504]) is sufficient for analysis; additional information is available in several printed books.

Table 10.1. Characteristics of cultural stereotypes according to Lewis' model, applied in WIS context

Linear-active	Multi-Active	Reactive
introvert	extrovert	introvert
patient	inpatient	patient
plans ahead	plans grand outline	looks at general principles
does one thing at a time	does several things at once	reacts
punctual	not punctual	punctual
compartmentalises activities	one activity influences another	sees whole picture
sticks to plans	changes plans	makes slight changes
sticks to facts	juggles facts	statements are promises
gets information from official sources	prefers oral information	information from official and oral sources
follows correct procedures	pulls strings	networks
completes action changes	completes human transactions	react to partners
likes fixed agendas	interrelates everything	thoughtful
uses memoranda	rarely writes memos	plans slowly
dislikes losing face	has ready excuses	must not lose face

10.2.3 Multidimensional Aspects of Culture

The individual level of culture is an intersection of overlapping cultures having their root in national, organisational aspects and a variety of sub-cultures (Figure 10.4).

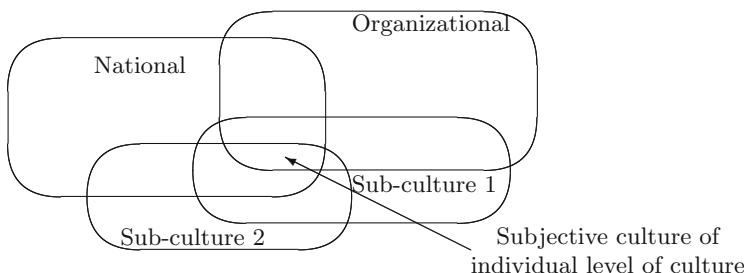


Fig. 10.4. Interrelated levels of culture

Figure 10.2 shows that the role of values is important especially in national culture level. When we take into account the factors modifying the national culture (Figures 10.4 and 10.1) the learnt practices and norms will become more dominant. These modified behavioural patterns have their roots in cultures coming from professional and organizational sources. These mod-

ifications are seen either in the form of permanently modified or situational cultural patterns.

E. Hall distinguishes between high and low context cultures according to the extent of “wordless” communication included in messages. The context gives additional information, which is necessary to encode the whole situation and background of a given information.

In *high context cultures* an information included in communication can have different meanings according to the context and it needs additional information to be understood, whereas in low context cultures an information included in the communication has only one single meaning. No additional information is necessary to understand it. In high context communication a message relates tightly to the communication context, which is expected to be understood by the parties. This indicates also collectivistic (group) culture, in which a variety of details are known by the group members without explicit messaging. The high context messaging includes hidden meanings, “facts between the lines”, culture-related sayings having certain commonly (collectivistic) accepted meaning.

The *low context* messaging instead is based on punctual clear messages having all information built in without possibilities to misunderstanding and wrong interpretations, even for those members of the audience not belonging to the society. From this point of view some similitude can be searched for between a high/low context communication and Hofstede's IDV based classification of cultures.

Tables 10.2 and **10.3** shows the differences between members of high or low context. In addition, **Table 10.3** lists some situations that are typical sources of misunderstandings and conflicts.

Table 10.2. Characteristics of high and low context cultures

	Low context	High context
Example countries	USA, UK, Canada, Germany, Denmark, Norway	Japan, China, Egypt, Saudi Arabia, France, Italy, Spain
Business outlook	Competitive	Cooperative
Work ethic	Task-oriented	Relationship-oriented
Work style	Individualistic	Team-oriented
Employees desires	Individual achievement	Team achievement
Relationship	Many. looser, short-term	Fewer, tighter, long-term
Decision process	Logical, linear, rule-oriented	Intuitive, relational
Communication	Verbal over non-verbal	None-verbal over verbal
Planning horizons	More explicit, written, formal	More implicit, oral, informal
Sense of time	Prenset/future-oriented	Deep respect for the past
View of change	Change over tradition	Tradition over change
Knowledge	Explicit, conscious	Implicit, not full conscious
Learning	Knowledge is transferable	Knowledge is situational

Table 10.3. How members of high and low contextual cultures see themselves and their opposites

How they see themselves what they are	High context communication	Low context communication
	<ul style="list-style-type: none"> • polite • respectful • integrates by similarity / harmony • indirect 	<ul style="list-style-type: none"> • open • true • integrates by authenticity • direct
How members of the opposite culture are seen	High context claims against low context	Low context claims against high context
	<ul style="list-style-type: none"> • impolite • ‘cannot read between the lines’ • naive • no self discipline • too fast 	<ul style="list-style-type: none"> • hiding information • not trustable • arrogant • too formal • too slow

Typical representatives of high context cultures come from Asia and Arab world, which also belong to collectivistic countries in Hofstede's classification. In the middle area of the continuum there are European countries, and in turn the low context cultures are represented by the North-European and Scandinavian countries. There seems to be a certain analogy between low context culture and Lewis' linear-active cultures.

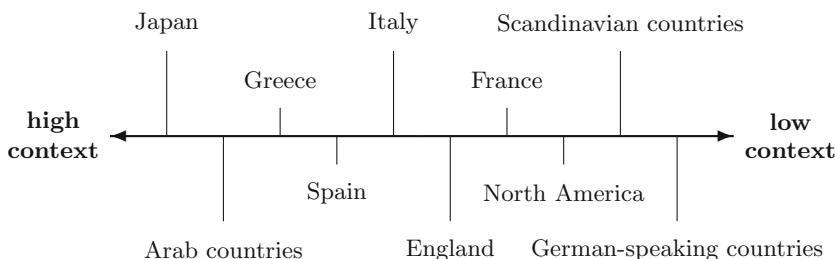


Fig. 10.5. High and low context cultures

None of the approaches covers all facets of culture-dominated behaviour and reception, but classifications can be combined to provide a more complete picture. Before combining them we have to harmonise the approaches. The Lewis model in Figure 10.1 essentially uses two or three differentiation rules:

Reception kind: Linear active people are oriented towards facts and are planning, whereas reactive people are rather receptive and responsive.

Faceted kind of work: People might prefer multi-tasking as multi-active people or might rather be oriented towards single-tasking.

Communication style: The communication can be more restrained or more emotional, impulsive or loquacious.

These differences are combined in the linear active (factual reception and single-tasking, potentially restrained), reactive (receptive and either single-tasking, potentially either loquacious or restrained), and multi-active (more responsive, multi-tasking, potentially more emotional) categories.

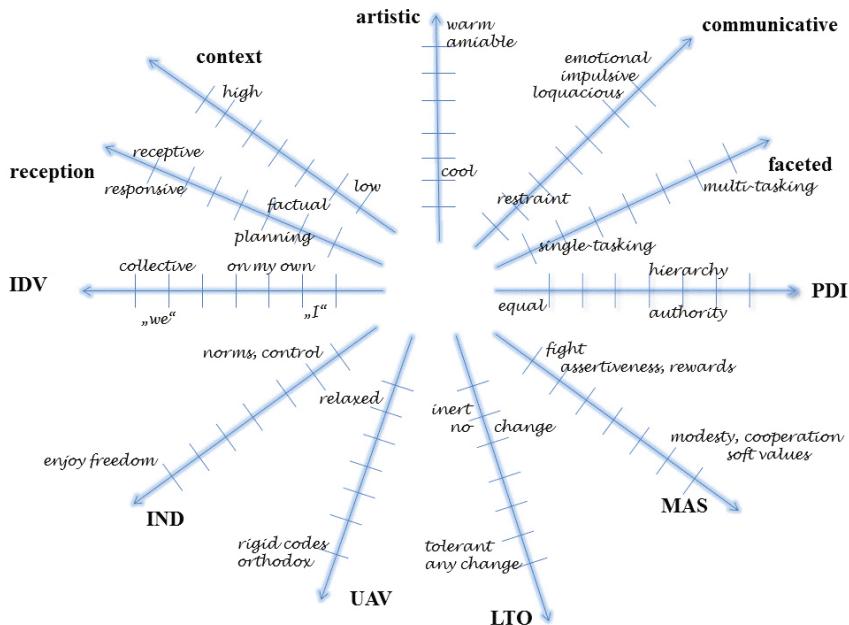


Fig. 10.6. The Kiviat graph of the cultural dimensions of people

10.2.4 Cultural Stereotypes and WIS

The comparison between cultures provides understanding of the differences in behavioural patterns between people representing different cultures. The source of stereotype analysis is in different communication and collaboration contexts. It can be used to avoid and solve conflicts in typical conflict sensitive situations. As already depicted in Figure 10.2, people's behaviour is guided, in addition to the national cultures, by several other cultural aspects, like organisational culture (habits adopted by the organisation), organisational subcultures and suborganisation cultures, work culture (similarities in

behaviour, interaction, decision-making, organisation structure, and goals), professional culture (education and adopted practices typical of certain professions), project culture (cross-section of organisational and professional culture) and team cultures.

Web design and screenography added to these dimensions the personality profile. This personality profile can be used to derive a number of orientations of people such as being businesslike or lyrical, conventional or inventive, classical or modern, traditional or vanguard, tough or tender, and rustic or cosmopolitan. We may combine specific orientations into pattern and general styles of web page styles. The most prominent general style considered in literature is based on the differentiation of the artistic style that ranges from cool to amiable.

We may also reorder the Hofstede dimensions according to their closeness. The resulting revision of the approaches with the main properties is displayed in [Figure 10.6](#).

We combine the approaches in a layered structure in [Figure 10.7](#). The idea of the compound model is to guide the user to take into account the different aspects needed in building / using information systems in multicultural context. In our earlier work we have analysed the importance of cultural issues in several IS functionalities: web information systems, information search, database structures and IS contents, communication, user interfaces and user preferences in services.

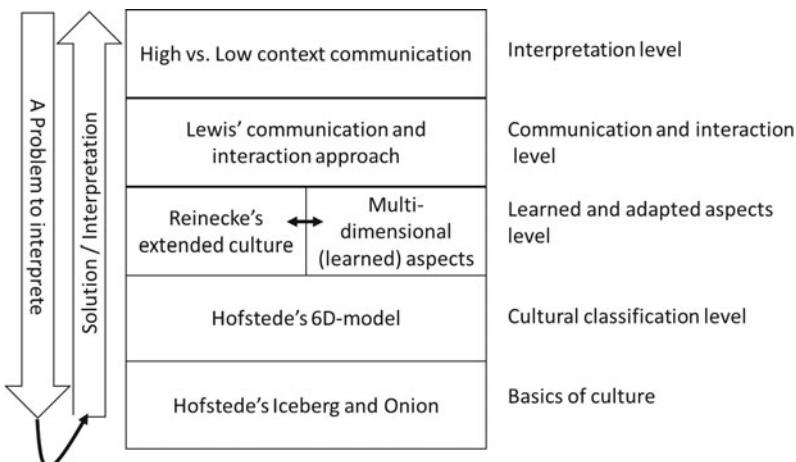


Fig. 10.7. A compound model for cultural analysis in IS context

10.3 Presentation Cultures

10.3.1 Deriving Guidelines for Presentation from Stereotypes

In [Table 10.4](#) we provide a short analysis of the possible cultural factors affecting IS usage. The diversity of roles has been discussed in Chapter 5 by refining the concept of user models. User models provide means for variations in behavioural patterns on a personal level, above the national culture level of the pyramid in [Figure 10.1](#). [Table 10.4](#) gives some answers to the question “how cultural differences of the users should be taken into account in information systems”.

Table 10.4. Cultural factors in information systems

IS Property	Culture analysis
Complexity of UI	More accepted in high PDI than low PDI cultures. Multi-active cultures are used in multi-tasking and are more familiar in handling tasks in non-linear manner.
Long response times	More accepted in high PDI than in low PDI cultures.
UI colors	Accepted and expected in multi-active cultures. Color map and meaning of colors differs between cultures. In some cultures colors have also emotional connections.
Symbols and logos	In principle symbols provide a common language. Symbols in different cultures may have different meaning. There are also symbols that are not proper to be used in some cultures.
Support for uncertain situations	High UAI cultures respect features that guarantee the correctness of operations - e.g., repeating questions and extra confirmation operations. In linear-active those may be disturbing.
Decision making	In low PDI cultures the users are more ready to make fast decisions that have wide influence. In high PDI cultures the users would need confirmation from colleagues in higher positions, which leads to circulation of the activity step by step. This kind of circulation is felt inconvenient in low PDI cultures.
Privacy issues	Individualistic cultures are more aware of privacy issues than collective cultures.
Feedback	Fast feedback is appreciated in cultures that have low LTO level; high LTO index indicates readiness to wait for the feedback.
Clarity	Low UAI cultures are tended to accept confused situations in IS usage; high UAI cultures appreciate clarity.
Predictability	Predictability of information system's behavior is expected in high UAI cultures.

Stereotypes of users can be combined into personae [588, 730]. A *persona* is characterised by an expressive name characterising the stereotype, by culture, by nationality, by organisations or teams, by a bundle of projects the

person might be engaged with, by a profession, by intents, by typical technical equipment, by behaviour pattern, by skills and profile, by disabilities, and by specific properties such as hobbies and habits. A persona is a typical individual created to describe the typical user, the context, the portfolio, and the profile. User models discussed below characterise profiles for education, work, and personality. This characterisation can be extended by an identity with name, pictures, etc., by personal characteristics such as age, gender, location, and socio-economic status, by a characterisation of reaction to possible user errors, by specific observed behaviour including skill sets, behavioural pattern, expertise and background, and by specific relationships, requirements, and expectations. A typical stereotype is the German Jack-of-all-trades that represents a specific kind of a business man in Germany.

User modelling is based on the specification of *user profiles* (see Chapter 5) that address the characterisation of the users and incorporate the stereotype of the user as default values, and the specification of *user portfolios* that describe the users' tasks and their involvement and collaboration. Chapter 5 distinguishes between the *education*, the *work* and the *personality* profiles.

A *portfolio* is determined by responsibilities and is based on a number of targets. Therefore, the user portfolio within an application is based on a set of tasks assigned to or intended by a user and for which he or she has the authority and control, and a description of involvement within the task solution [731]. A *task* as a piece of work is characterised by a problem statement, initial and target states, collaboration and presupposed profiles, auxiliary conditions and means for task completion. Tasks may consist of subtasks. Moreover, the task execution model defines what, when, how, by whom and with which data a task can be accomplished. The result of executing a task should present the final state as well as the satisfaction of target conditions.

10.3.2 Cultural Stereotypes, User Models and Information System Design

So far we derived the user model based on the cultural stereotypes. Classical information systems development is based on a three-layer architecture: the realisation layer, the conceptual layer and the user layer. The last layer is typically supported by a view set defined on top of the information system schema. We first develop a database schema and next derive a number of views on top of this schema. Using the schema and the views we may now apply transformation approaches for interpreting or translating the schema and the views to a realisation language, e.g., the data dictionary language provided by some object-relational DBMS.

This approach does not consider the culture or the user model. Based on our approach we derive now guidelines for the quality of the user interaction, guidelines for the development of the interfaces, and requests for adaptation of the system.

The view set must have also a clear and well-defined inner structure. This inner structure has to follow the logics of work and realise the specific requests according to the cultural factors in **Table 10.4** and the user model. We may restrict for user interaction the complexity of the user interface, the response time, the feedback time and the colouring schema. Furthermore, we may require specific support for complex views, for decision making, for privacy, clarity and predictability. For instance for users within most German regional cultures, a complex user interface and a long response time should be avoided. At the same time, each region has its specific culture for colouring, texturing and layout. Additionally, features for supporting discussions, delay structures for later completion, support functions for privacy preservation have to be provided. Furthermore, we have to apply a number of interface criteria such as clarity of the layout and predictability for the control flow.

A *guideline* for the quality of user interaction includes:

```

interface complexity (tolerated, not tolerated):
    (level 1, level 2)
support structure: view
support discussion: view
support functions: function
time behaviour response time: maximal tolerated
time behaviour feedback time: maximal tolerated
time behaviour storage time: automatic
privacy tactics: added system features
screen guideline: description
    colouring schema: description
    texturing schema: description
    screen quality criteria: description
workspace personal: view
    additional workspace personal: view
    shared workspace (which user, which): (user,view)

```

This list is not exhaustive. Nonetheless, it shows how such guidelines can be derived for certain user models, and it permits to satisfy typical requirements in relation to the personality of a user. It might be weakened in dependence on the education and work profile. It must however cover the portfolio of the user.

The user profile, the user portfolio and the corresponding stereotypes describe also implicitly the properties of interfaces, e.g., ordering of items, effects that support the work, and the layout and playout of screens. Such guidelines are based on the principle of *proper organisation* depending on the user model, on the principle of *economy*, e.g., non-redundancy of actions, on the principle of *collaboration* depending on the skills and abilities of the user, and on system design *standards*. For instance, most regions in Germany use a clear, predictable and well-organised structure both for data (i.e., with proper layout) and the actions (i.e., proper playout).

In a similar form we derive *guidelines* for the development of the interfaces:

- organisation structure:** pattern
- quality maximal redundancy:** level
- supporting abilities:** description
- support structure:** view
- supporting standards:** link
- layout guide:** description of preferred
- playout guide:** description of preferred

This list is also not exhaustive and can be extended depending on habits and traditions and others.

Finally, interfaces and systems must be adapted to the user culture, to the specific regional culture, to the organisation and to the users themselves. The system specification may result in a large number of views and functions. We will show in the sequel that this set can be reduced by a clever realisation. Views and workplaces can be far too large, especially if users work on singleton screens. Therefore, we need an automatic decomposition feature for these features. We may base this decompostion on the principle of closeness of items, i.e., closed items are not separated and distant items can be separated. In a similar form we can organise the flow of work in dependence on the linearity of activities, the kind of multi-activity, and the inner structure of process deployment of users.

This set of properties can be combined to *requests for adaptation of the system* that have to be performed during transformation to a realisation system:

- separation feature:** organiser
 - based on adhesion/cohesion: sub-structures
- harmonisation action set:** similarity level
- completeness action net:** closure condition
- supporting standards:** link
- quality conciseness:** organisation preferred
 - reactivity support: degree
 - multi-activity support: pattern
 - thought direction: linearity or network-oriented

This kind of adaptation can be supported by current technology, e.g., by refinements of the system, by a larger set of views that support the user, by a larger set of functions, and by workplaces supporting the user.

A Small Case Study

Systems for field staff are typical examples of cultural-dependent, multi-facetted, multi-structured and multi-functional systems. They must provide

many interfaces in order to satisfy all the demands. Database technology supports such systems by means of view towers, e.g., the incremental structure in Figure 10.8.¹

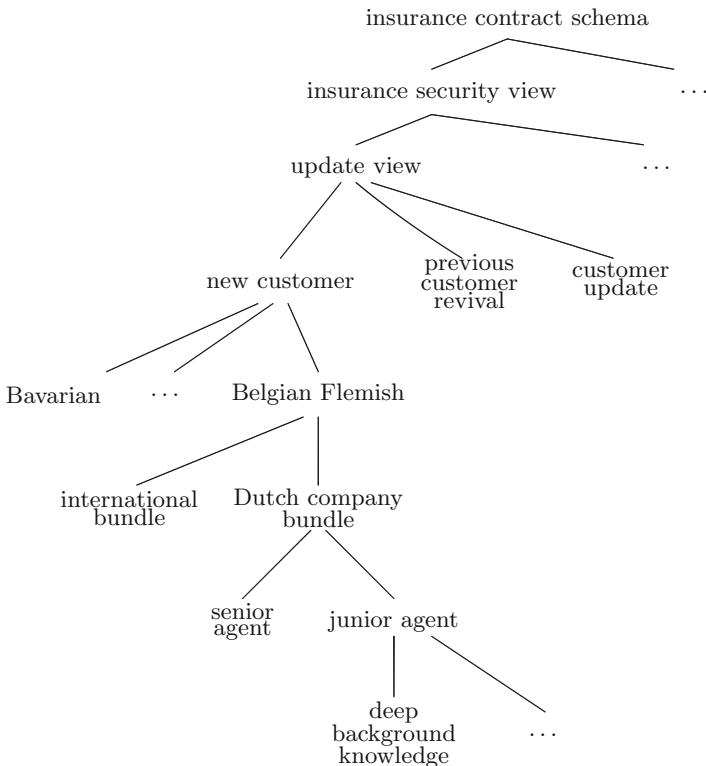


Fig. 10.8. A part of the view tower used in insurance applications for field staff agents

The final view of a junior field staff agent depends on the educational, work and personality profile of the given agent. The agent has to follow the organisational, project and team culture depending on the bundle of collaborating companies. Customer support depends on the region and the culture

¹ The views until level 3 have been developed in one of our industrial projects. During this project we realised that the development of views until level 7 would be very beneficial. Although the system developed is still in deployment, there are many requests for changes that might provide a better agent support within this application.

within the given region. The upper level views follow the typical structure of view-backed information systems. Work views enhance update views by additional data and features. Update views allow a direct update through this view in the given database. The data are guarded through the security views directly to the database which is structured according the database schema.

10.3.3 Cultural Stereotypes and Their Utilisation for System Development

The stereotype models (Hofstede, Lewis and others) are partially overlapping and indicate the same behaviour from different viewpoints. Reinecke's work combines user interface design with Hofstede's dimensions. Certain user interface properties and interaction process features have Hofstede index value. For instance, the layout of the web pages, color selection, guidelines for the interaction process, etc. are carefully bound in the interface properties (see [Table 10.5](#)).

Table 10.5. Relationships between Hofstede dimensions and UI design (modification of the work by Reinecke [669, 670])

Index	Low	High
PDI	<ul style="list-style-type: none"> · access variations, non-linear navigation · non-structured data allowed · most information at interface level, deep hierarchy not accepted · friendly messages · need for support is low 	<ul style="list-style-type: none"> · linear navigation, few links, minimal amount of alternatives · structured data expected · data in hierarchy, little data on top level · strict messages · support must be available
IDV	<ul style="list-style-type: none"> · traditional colors and images · image to text direction · high multimodality · colorful expression 	<ul style="list-style-type: none"> · colors used to encode information · text to image direction · low multimodality · monotonous color map
MAS	<ul style="list-style-type: none"> · little saturation, pastel colors · exploration and different navigation paths allowed · personal content presentation and friendly communication 	<ul style="list-style-type: none"> · high contrast, bright colors · restricted navigation possibilities · encouraging words in communication
UAI	<ul style="list-style-type: none"> · most information at interface level, complex interfaces · nonlinear navigation accepted · colours, typography and sound to maximize information 	<ul style="list-style-type: none"> · organize information hierarchically · linear navigation paths, guided · redundancy and cues to avoid ambiguity
LTO	<ul style="list-style-type: none"> · reduced information density · content structured into small units 	<ul style="list-style-type: none"> · most information at interface level · content can be arranged around a focal area

Reinecke's original table is based on collected information from several sources. It also contains one minor error in the PDI part, in which the last items are in wrong order.² In MOCCA the central role is played by the information system's understanding of the user in terms of user model. The user interface adaptation is based to the best fit between the user model and interface contents / interaction structure. This is worth a generalisation and transfer to another context, which is one of the aims of this chapter. The work of Lewis has not been taken up by Reinecke. Our aim is also to combine his ideas in our analysis.

The Kiviat graph in Figure 10.6 can be used for the derivation of style guidance for user interfaces and web pages. The Kiviat graph in Figure 10.9 uses the guidelines for the derivation of aspects of website presentations according to the specific culture.

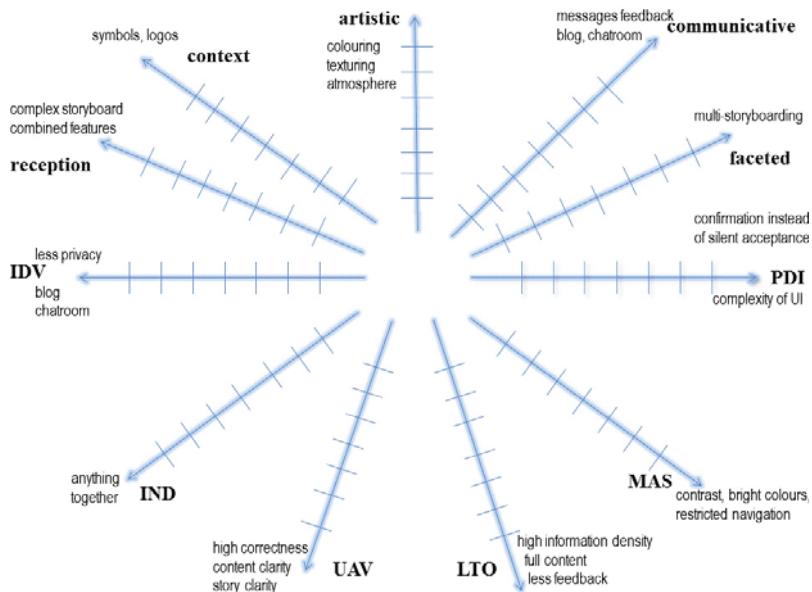


Fig. 10.9. The Kiviat graph for style guidance in dependence on the culture characterisation of people

The adaptation of website presentation is typically hidden and the secret of economically successful sites. Organisational (e.g., company, project, team) and regional (e.g., family, tribe) cultures are like country cultures. Never try to change one during website development. Instead, try to work with what

² However, this is relevant for our table and not included in it.

you have got. Here we concentrate on high and low context of cultures and set this approach in relation to our previous one with a focus on websites.

The Kiviat graph in [Figure 10.10](#) displays cultural differences of the three countries Finland, Germany and Japan. We use for Finland brown colour and a square dotted line, for Germany blue colour and a long dashed line and for Japan green colour and a solid line. The graph in combination with the style guidance in [Figure 10.9](#) gives a first style orientation for the design of web pages.

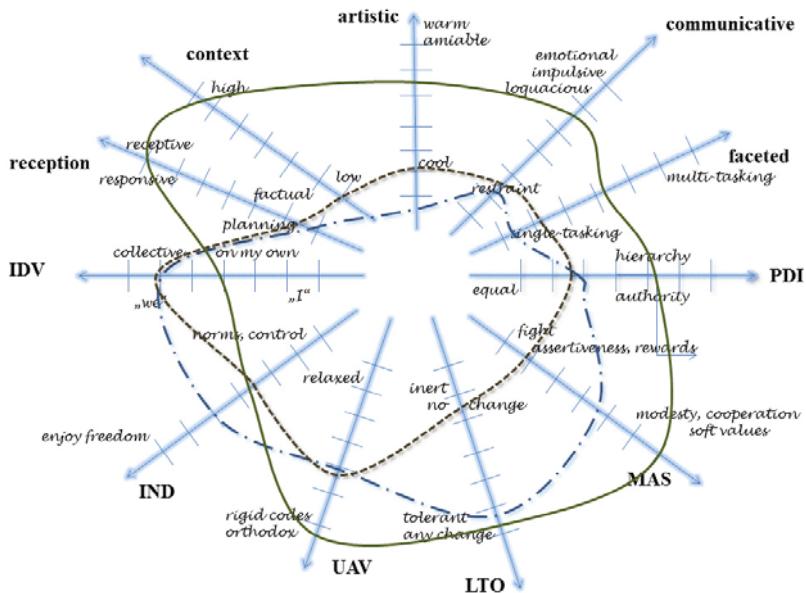


Fig. 10.10. The Kiviat graph for the three countries Finland, Germany, and Japan

10.4 Technologies for Realisation of Culture-Aware Systems

We introduce now an approach to generic and culture-adaptive storyboarding based on the mini-story approach [244, 871]. We observe [360] that database technology is fit for adaptation of content to the user culture. We introduce parametric database schemata and generalise the solutions presented in [17, 360, 361, 523, 528, 823]. The technical details are discussed in Chapter 11.

10.4.1 Culture-Aware Storyboards

We develop now a framework and a realisation strategy for automatic adaptation of storyboards to users. Mini-stories are the main constructive component for storyboards. A storyboard is a composition of mini-stories. They are generalised to parametric mini-stories. These parametric stories can be adapted to different cultures.

Mini-Stories

On a high level of abstraction the storyboard of a web information system specifies who will be using the system, in which way and for which goals. Storyboard pragmatics deals with the question what the storyboard means for its users. One part of pragmatics is concerned with usage analysis by means of life cases, user models and contexts. We also addressed another part of pragmatics that complements usage analysis by WIS portfolios. These comprise two parts: the information portfolio and the utilisation portfolio. The former one is concerned with information consumed and produced by the WIS users, which leads to content chunks. The latter one captures functionality requirements, which depend on the specific category the WIS belongs to.

The storyboard language we are going to use extends SiteLang [850] by explicitly introducing mini-stories. The conceptual model of storyboarding (see e.g., [727]) takes this up by providing an integrated model comprising the story space capturing the stories and the plot, actors, and tasks. Inspired by approaches in theatre and film, the story space comprises scenes and actions on these scenes, and the plot describes the details of the action scheme. Furthermore, the model describes actors in these scenes, i.e., groups of users, which leads to roles, profiles, goals, preferences, obligations and rights. The actors are linked to the story space by the means of tasks.

A *mini-story* [244, 871] typically captures a small, self-contained, tightly connected set of scenes similar to a clip for movies. It may be characterised through a (semantic) word field [202]. In B2C e-business, typical mini-stories are (1) advertise, (2) quote, (3) request, (4) response, (5) select and collect, (6) bargain, (7) contract, (8) requisition and order, (9) deliver, (10) invoice, (11) pay, and (12) return. Mini-stories have an input, an output, associated actors within certain roles and plays, controls for their playout, features to be used within the story, and conditions for their usage.

Parametric Mini-Stories

Similar to situations in applications, tasks may be specified on the basis of a general description of a possible way for satisfaction and completion. These activities may vary in their style and flow of scenes or activities. In traditional approaches, an activity is specified supported by its own mini-story, e.g., contract according to a specific set of regulations, contract in a private form or in

dependence of the parties, historical contracts, contract proposals, contract on the basis of other contracts, contracts according to company styles, contract as an extension, etc. The traditional approach will lead to a huge number of mini-stories that have a similar behaviour.

We extend mini-stories in order to become adaptable. The mini-story is independent of the parties involved, the application domain, the ends, the supporting means, the style and the pattern of applying the actions, the spatial and temporal context as well as other contexts, the content and the functionality to be used, the concrete intention or target of its utilisation, its benefits and values in applications, and the utilisation constraints. The dependence is expressed through an instantiation of corresponding parameters.

Parametric actions can also be annotated by verb fields that declare what kind of action is going to be used and what are the object and the valencies of objects within this action. Typical general parameters are: *wherefore, whereof, wherewith, worthiness, why, whereto, for when, for what reason, by whom, to whom, whatever, wherein, where, for what, wherefrom, whence, what, how, whereat, whereabout, whither, when, why, what properties, what scenario, which restrictions*.

These parameters generalise the rhetoric frame (who, what, when, where, why, in what way, by what means) that can be traced back to Hermagoras of Temnos [72]. These general parameters form an extensible *parameter space*.

Adaptation of Parametric Mini-Stories to Culture of Users

Parametric mini-stories can be adapted to the current situation, to the current user, to the systems that might be used, to the flow of parametric actions that seem to be appropriate, to the environment that is going to be used, to the styles and pattern, and to the conditions within an application. Other contexts are typically cultural contexts, e.g., layered structure [360, 411] (universal human nature, culture that is specific for a group, personality that has been inherited or learned) and multidimensional aspects of cultures [360] (national, regional, or specific cultures versus organisational, project, team cultures).

The Hofstede [327] and Trompenaars [868] dimensions govern the instantiation of parameters within the parameter space. We follow here the research on user interfaces (e.g., [670], Table 10.5) and presentation adaptation in [360] and generalise these results to stories.

- (I) People with high PDI score (or power close people) are used to being driven. So, the story may also be the driving force. Linear stories are better acknowledged. In cultures with low PDI values (or better “power distant people”)³ people prefer story spaces with many opportunities. Stories must not entirely structured.

³ In Hofstede’s terminology: low score in the power distance index.

- (II) Individual people accept also interleaved and multimodal stories. Less individual and more collective people prefer collaborative stories. At the same time, multimodality is lower.
- (III) Masculinity accepts also jumps in stories. Decisions must however be clear. Femininity requires more smooth stories with clear steps. Some decisions may also be delayed.
- (IV) Uncertainty avoiding people need a preview for next actions and next mini-stories. Stories should not interleave. So, we derive a story tree from our mini-story. Stories should be unambiguous. As the opposite, uncertainty tolerant people also accept story forests, i.e., interleaved, concurrent story trees that might reuse parts of others.
- (V) Long-term orientation requires that storyboards developed must become stable before being delivered to the user. They should not often be changed. Short-term oriented people prefer shorter stories. So, mini-stories can be kept on their own without integration into larger ones.
- (VI) Indulgence allows integration of storyboards with others. Conservative or restrained people cope better with flow of events they are used to.

The Lewis separation into linear, reactive and multi-active cultures (see [360, 502, 504]) guides the adaptation rules and the parameters as well. Linear-active culture is better supported by a sequential story flow whereas multi-active culture allows to derive multi-tasking, parallel flows of actions. Stories for reactive people are better given in a fully fledged form with zoom facilities for current activities.

10.4.2 Culture-Aware Content

We develop now a framework and a realisation strategy for automatic adaptation of content to user cultures. Classical information systems are build on top of a database system that is structurally specified on the basis of a conceptual or logical or physical database schema. Views provide a facility for derivation of user-specific data. We generalise this approach by introducing parametric database schemata based on pattern from [17] and by developing general views that are considered in detail in Chapter 11. These views can be adapted to the user culture through generation of specific views for each stereotype of culture. The concept of view towers in Chapter 11 has already been used for generation of interfaces [360].

Parametric Database Schemata

Database schemata can conceptually be represented by diagrams in an extended entity-relationship model [823]. These schemata reflect however some specific viewpoint and some culture. They are dependent on assumptions, paradigms and postulates that drive development of such diagrams. Linear-active culture is neatly supported by development tactics such as Salami slice

or class-oriented schemata. Reactive culture is better supported by schemata that reflect all aspects of a given thing from reality, e.g., by XML schemata (networks). Multi-active culture is however not reflected at all. In order to support this culture, a large number of views must be developed. Therefore, we need a more flexible way for representation of a database structure.

A *parametric database schema* consists of a collection of structure patterns together with a composition that binds some of the parameters of the pattern to each other. Patterns are parameterised. Parametrisation also uses a parameter space similar to the one for mini-stories. The composition of pattern follows the approach developed in [523, 528]. In general, it is an open database schema with parameters that can be adapted according to the specific form of the web information system. A structure pattern [17] consists of a name of the pattern N , the schema of the pattern S , deployment conditions Ψ , parametric integrity constraints Σ , the parameters p_i of the pattern with their pre- and post-conditions γ_i, δ_i , the controls \mathfrak{C} for all functions that can be applied to the pattern, and the supports \mathfrak{U} for all functions that can be applied to the pattern, i.e.,

$$\mathcal{SP} = (N, S, \Psi, \Sigma, \mathfrak{P}, \mathfrak{C}, \mathfrak{U}).$$

We can now develop a number of *stereotypes for database schemata*:

- (a) strictly hierarchical (ER-like) database schemata,
- (b) schemata with local viewpoints that reflect the needs of some stakeholders (local-as-view approach),
- (c) variants of XML-schemata, Bachman diagrams,
- (d) sets of local database schemata with the requirement that the corresponding database schemata is simply the union of the set (global-as-view based on local viewpoints),
- (e) sets of personalised views based on local database schemata with some kind of coherence constraint among all views (rigid global-as-view), etc.

This list of schema stereotypes is not exhaustive. It demonstrates however that different schemata can be given for the same application. Which database schema is appropriate depends on the utilisation of the database schema and on the community of practice and their culture.

Parametric Views

Similar to parametric database schemata we can derive parametric views. The set of views can form a view tower [360]. Views are schemata on their own,⁴ e.g., alike the view in [Figure 10.11](#).

⁴ Classical object-relational approaches only support singleton table views. In this case we must define a complex view as a collection of views that are associated through integrity constraints - mainly (pairwise) (generalised) inclusion constraints.

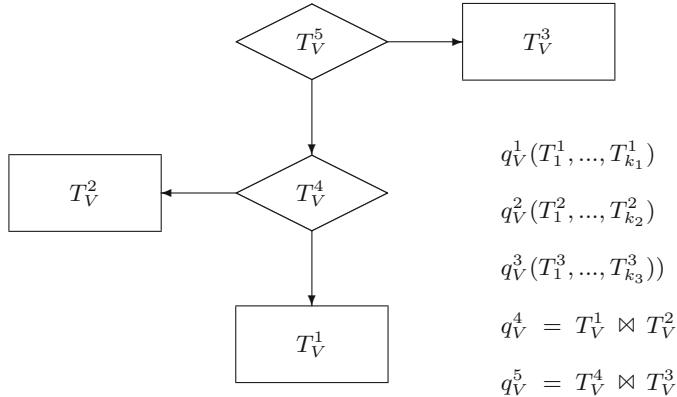


Fig. 10.11. An abstract example of a view in the higher-order entity-relationship model with five types T_V^i that are defined five queries q_V^i which use base types $T_1^1, \dots, T_{k_1}^1, T_1^2, \dots, T_{k_2}^2, T_1^3, \dots, T_{k_3}^3$ where the two relationship types are defined through joins of their component types

We use the algebra of the higher-order entity-relationship model [823] for view specification, i.e., union, difference, intersection, projection, join, selection, renaming, filter, aggregation, variation for presentation, data presentation (sorting, grouping, etc.), nest, and unnest operations. Additionally [360], views are associated

- with actions and users (e.g., for authorisation),
- with enforcement control for modification of data through views, and
- with data maintenance policies (either virtual views as a specific computation formula with query expansion techniques or as materialised views with an enforcement policy for the case that data in the base database tables are modified).

A view inherits the parametrisation of types that are used in their query expressions. There is no need to unify the parameters since these parameters are identifiable by the query that uses them.

Adaptation of Content Schemata to the Culture of Users

A *content schema* of a web information system consists of a database schema and a collection of views. Views can be enhanced by functionality. The database schema and the views are typically bound by data enforcement strategies. The classical enforcement is based on a rigid local-as-virtual-view approach, i.e., data for views are virtual and generated only when there is need of them. Data warehouses follow the partial local-as-materialised-view approach, i.e., some of the views are materialised and are going to be changed whenever the (global) database is updating correspondingly. We may also use

the global-as-view approach where the local databases are the master and the global database is the slave that follows the local ones.

A content schema is *integrated* with parametric mini-stories such that each mini-story action only consumes data according to the content schema (e.g., one of its views) and produces only data that can be used for a modification of the content database (i.e., either for one of its updateable views or for the database itself).

Database schema stereotypes may be classified in a form similar to the classification of database schemata discussed above ((a)-(e)). Cultures are stereotyped as a combination of [327, 502, 503, 868]. Which kind of diagram is used also depends on the educational and organisational cultures within the community of practice.

Database schema stereotypes can be associated with cultural stereotypes as shown in [Table 10.6](#). We do not intend to present all possible aspects but use it as an example. Our table is based on [670] (and [Table 10.5](#)) but concentrates on structures instead of interfaces. The adaptation mechanism which can be used for this flexibility uses the database schema translation approach [823]. We start with a database schema and general views. The translation rules depend on the stereotype for the content schema.

10.4.3 Culture-Aware Functionality: Search

Information Searching Is a Difficult Issue

Users often face the problem that their mental model and their fact space are insufficient to answer more complex questions. Therefore, they seek information in their environment, e.g., from systems that are available. Information is data that have been shaped into a form that is meaningful and useful for human beings. Information consists of data that are represented in form that is useful and significant for a group of humans. This information search is based on their *information need*, i.e., a perceived lack of some information that is desirable or useful. The information is used to derive the current *information demand*, i.e., information that is missing, unknown, necessary for task completion, and directly requested. It is thus related to the task portfolio under consideration and to the intents.

There are various notions of information. We use here the anthropomorphical or pragmatic notion [845]: *Information as processed by humans, is data perceived or noticed, selected and organised by its receiver, because of his subjective human interests, originating from the user's instincts, feelings, experience, intuition, common sense, values, beliefs, personal knowledge, or wisdom simultaneously processed by his cognitive and mental processes, and seamlessly integrated in his recallable knowledge.* The information demand of a user can be supported by systems such as information systems, database systems, and information retrieval systems. In these cases a user issues a search query to

Table 10.6. Cultural stereotypes, kinds of database schemata that are potentially preferred, and potentially useful database schema stereotypes

Cultural stereotype	Preferences	Schema
High Power Distance	completely specified and well-formed, easy to understand and persistent database schema	(a)
Low Power Distance	freely configurable database schemata that is adaptable to current needs and preferences	(d)
Individualism	my own database schema according to my and only my preferences (work profile, education profile, personality profile, security profile)	(e)
Collectivism	commonly agreed database schema reflecting all elements within a group according to the collaboration style	(b)
Masculinity	restriction to essential elements and only those, strict structuring	(a)
Femininity	schema with additional and optional elements, with exploration opportunities, personalised schemata	(e)
Uncertainty avoidance	complete schema with all elements, hierarchical structuring, more linear, well-scoped sub-schemata with simple reference to main schema	(a),(d)
Uncertainty tolerance	extensible schema, flexible schema style, web-like schemata	(c),(e)
Long-term culture	all potential elements are reflected as well as all viewpoints, focused (oil stain) schemata	(a), (b)
Short term culture	handy schemata depending on current use and smooth integration of them, decomposable schemata	(e)
Indulgence	schema with a central part containing all necessary elements and further elements that might be of use in future	(e),(c)
Restraint	puritanical schemata without any non-essential elements	(a)
Linear-active culture	schemata with step-wise exploration of all its aspects	(b)
Multi-active culture	different variants of the global schema for parallel integrated work	(d),(c)
Reactive culture	completely fledged schemata with all details and views for later work	(d)

the system. Querying is however only one kind of searching. We know at least six other kinds of search [72] that should be supported:

- (1) seeking data by browsing, understanding and refining;
- (2) property-based questioning;
- (3) ferreting out data necessary by discovering;
- (4) searching by associations and drilling down;
- (5) casting about and digging into the data;

- (6) zapping through data sets based on search techniques, e.g., uninformed search).

Users need thus (α) best-suited content, (β) in an understandable presentation, (γ) in an easy form to overlook, (δ) at the right amount, (ϵ) just in time, (ζ) just for current payment, (η) to the right place, and (θ) to the right device.

The Classical ETL Approach

The classical Extract-Transform-Load model to information extraction from a database or information system follows the cycle represented in [Figure 10.12](#). Somebody faces a problem and wants to know some answer. The seeker realises that something is not understood as it should be. So, the user becomes aware of an information need. Next, an information demand is developed. Based on the knowledge about the information system the user might now ask a number of questions to the system. These questions must follow the style of the system. The system delivers an answer that might be understood by the user. The user integrates this answer into the mental models he/she uses and evaluates the answer according the information needs.

This approach to information seeking is currently used for search engines such as Google or Yahoo and also for directed search [360, 362]. Google search is based on keyword search sequences and requires from the user good preparation in the case of complex search requests. Yahoo allows a collection-based search. The user is supported in this case by a more sophisticated question generation and a better answer.

Open issues are currently however:

- How a user can proceed in a more efficient and effective way while satisfying the information demand?
- Users are very different. Is there a cultural difference between the users? How this cultural difference can be supported in a better form?
- What technical means can be provided for such support?
- Can we use the knowledge on the culture and derive culture-governed support for each of the users?

We will approach this topic by giving the (partial) answers to the questions from two directions. One is to analyze the behavior of IS users in different cultures. The main problem is to find a good fit between the user culture and the IS culture.

The Kernel: Understanding the Information Demand

Information consists of data that are represented in such a form that is useful and significant for a human or for a group of humans. Information need refers to a perceived lack of something desirable or useful. Information demand

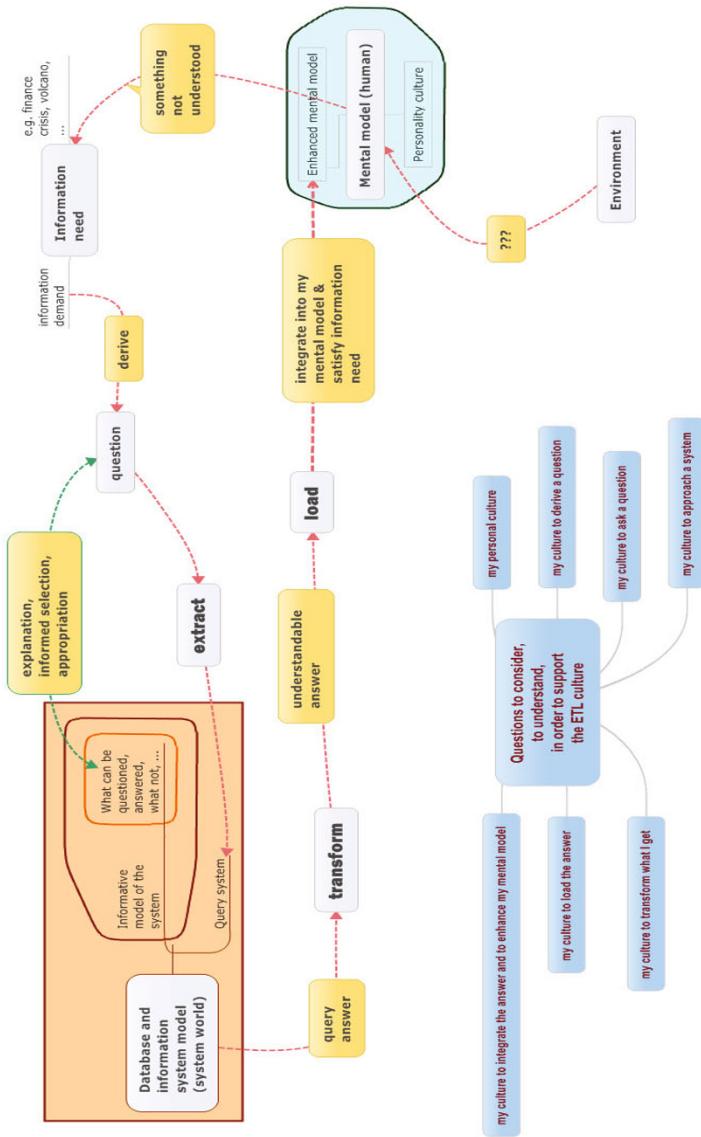


Fig. 10.12. The classical approach to information seeking through a database or information system

results from an act of demanding or asking. The *information need* is generally related to objectives such as becoming informed. The *information demand* is related to the portfolio under consideration and to the intents.

Information logistics that is built into a system can be the technical means for satisfying the information demand. Information logistics orients on best-suited content in an understandable and easy to survey form at the right amount just in time just for current payment to the right place and device.

Users typically request or need various content depending on their situation, on material available, on the actual information demand, on data already currently available and on technical equipment and channels on hand. This request is driven by the profile and portfolio of the user, by the national and regional culture, by the organisational and other cultures. Their (rough) profile and portfolio [360] can be used to derive the information demand of the user. This information demand is an orthogonal and implicit dimension of the request. Therefore, we need a facility for content and function adaptation depending on the culture and context of the user. Content adaptation and function adaptation may be thus considered as one of the ‘grand’ challenges of modern internet.

In information search a user is solving a concrete problem that needs data to be solved. The data is stored in data repositories – some of them are structured (like databases) and some not organized (like web based data). The information search is organized by the user into the form that has source in his or her mental model. In this mental model one central factor is the user’s culture.

In turn, the structure of the stored data is based on many factors – part of them are technical and part also based on the “cultural factors” of data repositories. The term “data culture” covers such elements as what data is stored, what is the conceptual model used in organizing the data, how the access to data organized, etc. In business data the culture follows the business needs, but especially in open data national (administrative) culture aspects may have an important role as a part of the data culture.

The user interface acts as a mediator between the user needs and the data repository. On the user side it adapts data repository to meet the needs of the user. On the data repository side it adapts user needs to fit on the services provided by the repository.

To simplify our approach, information search is based on communication in multicultural context – cultures of users and cultures on the background of data repositories. User interface is a mediator between that adapts the needs to the service opportunities. In our work we separate two categories of user interfaces. The first one, traditional information systems are usually implemented according to the needs of the different interest groups. In principle the fit between the needs and service capability provided by the data repositories should be good. In a globalised world, however, this is not anymore true. Software is developed for global market and it adapts only in limited amount to the local needs. The second category covers open access to different data sources - access via Internet, open access data, etc. In this situation the interest groups are not known to the data owner and needs of different interest groups are impossible to know and to take into account. In this context a

typical approach in information search is to use search engines as a mediator. The culture sensitivity related aspects in connection with search engines was handled by the authors in their paper [362].

Questions Answered During Information Search

Information search can be underpinned by questions the user might ask. We use the *W*H framework* [175] for questions such as: Who, what, when, where, why, in what way, by what means, wherefore (purpose), whereof (origin), wherewith (carrier), worthiness ((surplus) value), by whom, to whom, whichever, wherein, where, for what, wherefrom, whence, what, how, why, whereto, when, for which reason, whereat, whereabouts, whither, when. Alternatively, according to [70, 286] we distinguish between

simple/shallow questions in the form of verification (yes or no answer), disjunctives (is X , Y , or Z the case), concept completion (who? what? when? where?) or example search;

intermediate questions look for feature specification (what are the properties of X ?), quantification (how much? how many?), definition (what does X mean?), or comparison (how is X similar to Y ?);

complex/deep questions seek an answer by interpretation (what does X mean?), causal antecedent (why/how did X occur?), causal consequence (what next? what if?), goal orientation (why did an agent do X ?), instrumental/procedural (how did an agent do X ?), enablement (what enabled X to occur?), expectation (why did X not occur?), or judgmental (what do you think of X ?).

Simple questions directly follow the W*H framework. Intermediate questions can be supported as long we can express them on the basis of the W*H framework. It seems to be impossible to provide a sophisticated technique for complex questions. Therefore, we concentrate on the first two types. Each of the W*H questions is directed to the expected answer type. We can, for instance, relate the questions of the rhetoric frame as follows:

who: person in a variety of roles, party, institution, government, organisation,
... in a variety of roles,

what: product, service, material, procedures, techniques, ...

when: time of different kinds and contexts,

where: locations in various contexts,

why: transaction, activities, ...

how: document, bill, record, contract, agreement, data sets, ...

whereby: system, infrastructure, ...

Self-organisation for questioning is based on the notion of the kind of information demand:

Information demand because of data deficits occurs when there is an obstacle to a goal, a contradiction, an anomalous event, a glitch in an explanation, an obvious gap in knowledge, or a decision required between equally attractive alternatives.

Information demand that is monitoring common ground orients to gauge, to assess, to confirm, or to ratify what can be obtained from a database about a topic.

The first group is defined as *information seeking questions*. Other kinds of questions besides the two groups need very sophisticated support and are out of the scope in this chapter. Questions that are driven by *social coordination of action* include in-direct requests, indirect advice, requests for permission, and moves in bargaining. Questions that orient on *control of conversation and attention* such as greetings, directives to change the speaker, rhetorical questions, or gripes are not considered.

Query-Answer Forms for Information Searching

Answers to information seeking questions use concepts that can be generated from the information system. We based our approach on solutions developed in [72]. Information seeking questions to a system are given by the nested quadruple

(question content, matter (concepts, situation), user (profile, portfolio), carrier).

Answers to such questions that are expected from a system are given by the nested pair

(answer content, solution (characteristics, context, value)).

According to [72], a question has

- a *matter* (what, concepts, in what way) and
- a *situational context* (when, where, by what means).

A user has

- a *profile* (who) and
- a *portfolio* (wherefore, wherein, where, for what, wherefrom, whence, what).

He or she uses

- a *carrier language* (wherewith) within a certain *namespace* (whereto, by what means).

The answer expected can be characterised by

- the *answer characteristics* (how, why, whereto, when, for which reason),
- the *answer context embedding* (whereat, whereabout, whither, when), and
- the expected *surplus value* (worthiness) of the answer.

Matter, context, profile, portfolio, language and acceptance of answers depend on the general characteristics of users and therefore on their cultures. We shall see in the sequel that cultural stereotypes can be used for proposing default values for W*H framework:

- Cultures determine a partial answer to:
 - the matter (what, concepts, in what way),
 - the situational context (when, where, by what means), and
 - the answer context embedding (whereat, whereabout, whither, when).
- Cultures determine the profile (who), portfolio (wherefore, wherein, where, for what, wherefrom, whence, what) and the carrier language to a larger part.

Users want to satisfy their information demand either because of data deficits or by monitoring common grounds. We may support shallow or deep questions. But this seldom results in a singleton information request to a system. Users in different cultures also have learned different *tactics* how to satisfy their information demand. These tactics can be broadly categorised into *scheduled* or *straightforward* tactics. Typical straightforward tactics are depth-first, breath-first, adhesion-directed, or metaphor-based tactics. Scheduled tactics are, for instance, those that have been used in Artificial Intelligence search methods [569] such as hill-climbing, beam-search, best-first, simulated annealing, tabu search, branch-and-bound, alpha-beta search, etc. We may also categorise tactics by answer organisation such as *ordered answers* or *answer spaces*.

Communication Aspects Concluded

The findings of the analysis above are collected in [Table 10.7](#). The aim is to list typical information search related features structured in four viewpoints:

- Sources used in opinion building (*what* sources are used);
- The features of the communication pattern in information search (*how to search*);
- The reactions of the user in responses (*how to react* in the response);
- Structure of the search case (*interaction tactics*).

The items (some most important ones) are picked up from the introduction above and clustered to point out the culture related differences in information search context.

[Table 10.7](#) heavily simplifies the communication situation and focuses on few (selected) aspects in information search context. In a global environment

Table 10.7. Culture related aspects in information search context

	Linear-active, Data oriented	Reactive, Listener oriented	Multi-active, Dialogue oriented
What	Facts; Trusted persons; Prepared in advance; Expert opinions	Web sources important; Personal contacts used; Collectively prepared; Expert opinions	Informal networks important; Personal contacts important; Several moves ahead planned; Suspicious sources (subjective)
How / search	Simple linear structure; Direct access; Simple UI; Privacy appreciated	Partitioning, parallel; Several sources; Complex UI; Collective ownership	Iterative, extensive; Partitioned, parallel; Conservative; Purpose directed
How / re-response	Direct reactions; Facts only noticed; Details extended; Technical exact data	Analyzing, reacting; Details recognized; Collectively decided; Visualization, images	Suspicion on the reliability; Purpose directed interpretation; Interaction; Variety
Tactics	Direct, Linear, Logical; Step by step; Predefined agenda; Finish at once	Structured, complex; Parallel; Reactive agenda; Solved in chunks	Hierarchical; In pieces, convergent; Chess player approach; Tasks, random order

an important factor is also the language used in communication: native language vs. foreign language.

Let us consider a question-answer structure that consists of:

- *information seeking questions* in a system are structured as a nested quadruple (question content, matter (concepts, situation), user (profile, portfolio), carrier) and
- *answers to such questions* that are expected from a system are given by the nested tuple (answer content, solution (characteristics, context, value)).

We can conclude for this structure query-answer forms in [Table 10.8](#).

The Profile of an Information System

A user needs to know whether an information system is appropriate for the current information demand. This characterisation can be given on the basis of an *informative model* [848] of the given system. The informative model consists of the cargo of the system, a description of its adequacy and dependability, and evaluations of the system. The cargo is typically a very general instrument insert like the package insert in pharmacy or an enclosed label. It describes the system, the main functions, the forbidden usages, the specific values of the system, and the context for its usage. Following [541, 854] we describe the cargo by a description of the *mission* of the system in the usage scenarios, the *determination* of the system, an *abstract declaration of the*

Table 10.8. Culture sensitive query-answer forms

	Linear-active, Data oriented	Reactive, Listener oriented	Multi-active, Dialogue oriented
QUESTION			
Question content	Well organized, ranked sources, small selection of alternatives	Partitioned, organized in a flexible way, wide selection of alternatives	Incremental, iterative, situational sources
Matter (concept, situation)	Data driven concepts, preplanned situation	Changing and reactive both concepts and situation	Preplanned situation, collective conceptual structure
User (profile, portfolio)	Individual	Collective, authorities and hierarchies	Collective, flexible hierarchies
Carrier language	Linear, simple	Visual, complex	Varying
ANSWER			
Content	Text, data, all aspects covered	Visualisations preferred, rich content, gaps filled	Based on expected data, gaps accepted and freely interpreted
Solution (characteristics, content, value)	Organized, stepwise solution finding	Collected from pieces	Flexible interpretations, incremental

meaning of the system, and a narrative explanation of the *identity* of the system. The informative model of the system informs a potential users through bringing facts to somebody's attention, provides these facts in an appropriate form according their information demand, guides them by steering and directing, and leads them by changing the information stage and level. Based on the informative model, the user selects the origin for usage with full informed consent or refuses to use it. It is similar to an instruction leaflet provided with instruments we use.

One approach for developing such informative models is system profiling [509]. Profiling is the application of data analysis techniques to existing systems for the purpose of determining the actual content, structure, and quality of the data. We thus describe properties of the information system:

- the structure, the query facilities, the access restrictions,
- accurateness, completeness and currency of data,
- metadata, the information system history, and
- provenance characterising the usage of the system.

Profiles also might include a description of limitations of the system and its data.

The profile and the informative model of the system provide thus an intuitive understanding, of the restrictions, of the benefit and of the drawbacks of the system. It informs a potential users through bringing facts to somebody's attention, provides these facts in an appropriate form according their information demand, guides them by steering and directing, and leads them for usage of the system.

Input and Output Forms

An *input form* consists of

- a question word field* with valences as parameters of the question,
- a pattern* for structuring the input such as a menu or graph form,
- a collection of database views* that allow to clarify the matter of the information seeking question,
- a protocol* for processing the request,
- controllers* for processing the information request,
- supporting means* and methods for delivering the information request to a system, and
- defaults* for all parameters for the case that a user does not specify values for the parameters.

Defaults allow a quicker selection of the corresponding input form. The input form is dependent on the user profile and the user portfolio. Components for session management are typical supporting means. The session manager provides functions for opening, logging, and closing of a session, for history treatment, for security and privacy, for import of user data, and for handling of volatile data.

We might thus use the IDEF graphical language for depicting such input forms, e.g., in [Figure 10.13](#).

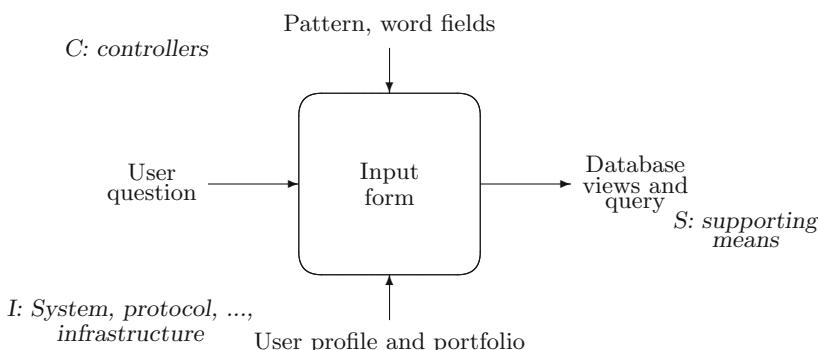


Fig. 10.13. A graphical icon representing an input form

An example of such input form is the following one:

Active input form: in parallel

```
open(content base); inform(proprietor, usage)
```

Self-protecting view schema: content based on collection of views with a protocol:

```
contact(proprietor, possessor, usage); obtain(proprietor, token);  
provide(media type, token)
```

Communication protocols based on service (distributed ADT), signals, shared variables, sender/reponder ASM (signature (e.g., signal), phases (via small ASM)), and timer represented through SDL or message sequence chart or other protocols

Security techniques against passive/active sniffing, trust exploitation, viruses, downloadables, OS holes, hacking

Control techniques for focusing, access, user authorization, (password) protection, biometrics, content/concept/topic security, firewalls, hiding/anonymizing/translating, identity management

Privacy enhancing techniques based on virtual private networks, key encryption, secure transaction, and corporate policy on security, privacy and control, use of cookies, etc.

A user can use a good number of input forms. Whether one input form or a bundle is used depends on the the profile and portfolio of the user, e.g., on the specific culture.

The output form corresponds to some load feature in the ETL approach. Users typically prefer some style of answers. The style however might vary a lot. Styles generalise the docket approach and envelopes developed for SOAP services.

The *output form* consists of

an answer style that supports structuring of the answer,

an answer collection of database views that allows to deliver the answer to the user from some system,

a protocol for delivering the answer and for tracking on the metadata on answer delivery,

controllers for the processing the information request,

a metadata and context pattern for declaration of provenance, temporal, quality, source, provider, data usage policy,

supporting means for deployment of the answer, e.g., for integrating the data, for delivering the data to the user, for associating and annotating the data, for restructuring the data, for storage and recording the data, for tracking history and logs of the data, for working with the data, and

defaults for all parameters for the case that a user does not specify values for the parameters.

We can use a similar graphical or iconic representation of the answer form. Also, various specific answer forms can be developed. Answer styles may use tabular, zoned, graphic or narrative formats.

The Enhancement of Storyboards

Storyboarding in a process-oriented holistic manner focusses on user intentions. The conceptual model of storyboarding (see e.g., [850]) takes this up by providing an integrated model comprising the story space capturing the stories and the plot, actors, and tasks. This approach has already been used for derivation of culture-based checkout mini-stories in e-business in [362].

A typical mini-story is displayed in [Figure 10.14](#). A customer may start without any selection or continue shopping with the given basket or add a product due to another selection. Then the catalogue is considered. This consideration may end the mini-story or may be continued by a configuration of the product. We see already with this mini-story that the two options have similar opportunities for leaving the mini-story. Therefore, it seems to be reasonable to add additional operators that can be applied to generic scenes. These operators have already been investigated for component-based generic database schema construction. A generic scene uses parameters that have either been used for scene characterisation or for the context to other scenes.

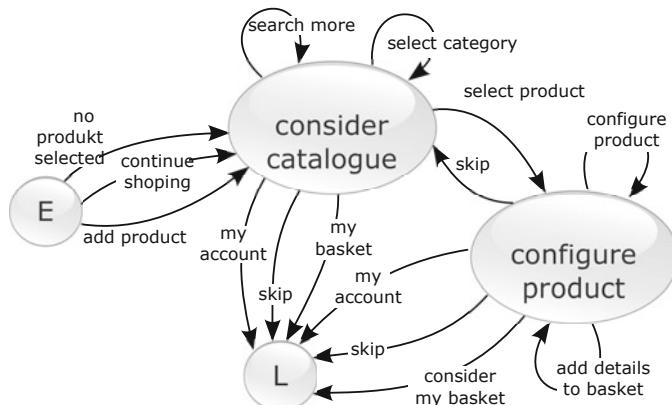


Fig. 10.14. The product selection procedure

In the given scene we have a working scene that is mapped to consideration of the catalogue or configuration of a product.

```

work_scene := (activity, links_within_activity, links_to_next)
configure_product_scene := REFINE(work_scene,
    links_within_activity -> add_details_to_basket
    ∨ consider_product
    links_to_next -> (consider_catalogue; skip),
    (l(eave); my_account ∨ skip ∨ consider_my_basket ) )
  
```

```

configure_catalogue_scene := REFINE(work_scene,
    links_within_activity —> search_more ∨ select_category
    links_to_next —> (configure.product; select.product),
        (l(eave); my_account ∨ skip ∨ my_basket ) )

```

The refinement is given by an assignment system. Each of the elements in this equation system is either defined in the previous step or in another refinement within the given system. This approach is hierarchical and thus usable in the same way as mathematics uses systems of differential equations.

We may now adapt the mini-story to a specific culture, e.g., for German culture (1) by adding extended facetted search and product comparison scenes and (2) by adding filters and corresponding addition/deletion actions.

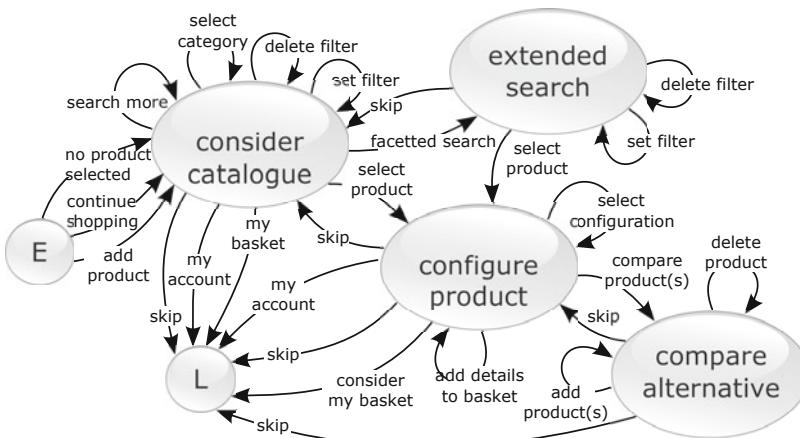


Fig. 10.15. The extended product selection procedure for Germans

The German variant of the same mini-story is displayed in [Figure 10.15](#). We apply a similar refinement step with the *REFINE* operator for derivation of the two scenes as a refinement of the *work_scene* and an *ADD_LINK* operator for additional links. In a similar way we also may use *DELETE_LINK* and *UPDATE_LINK* for a given scene.

Refinement of mini-stories and stories also uses *SPLIT* operations for decomposition of (mini-)stories and *COMBINE* operations with side (mini-)stories that provide a feature for addition of parallel (mini-)stories. Scenes in the storyboard have also a parameter for conditions (for entering and leaving the scene, for working within the scene), for input and output forms, for actors that may use the scene, and for functionality provided with the scene.

10.5 Bibliographical Remarks

Our approach to adaptation of WIS to the user culture is based on [354, 360, 362, 363, 583, 580, 226, 730, 852, 851]. In [362] the ideas of Reinecke [669, 670] have been adopted, modified and related to the concept of an *extended* or *adapted culture* of an individual. Especially, a WIS must satisfy a super-requirement: it must be an adaptable [230] and an easy to use system without any additional learning effort. Website development approaches are used to emphasise the ease of use [720] as a generalisation of ergonomics [54] and usability engineering [611]: usable without additional training, simple operating, obvious operating, simple for everybody, straightforward, within expectation, with context-sensitive help, with adaptable selection of wording, and with simple dialogues. Dynamic systems [666] react to changes in their environment in a quick and flexible way. Our approach is based on the culture research by [323, 324, 325, 326, 327, 411, 502, 503, 504, 505, 667, 668].

National culture (language, educational tradition, religion, beliefs, attitudes, and social context), *regional cultures* (as a specialisation of national cultures), *organisational (work) culture* within an organisation, *professional culture* based on education and adopted practices, and *project culture* or *team culture*. Duzi et al. [205] extend this categorization with team cultures. The more detailed review of different aspects in the concept of “culture” is given in [353]. These models of culture can be related to user models [230]. The multidimensional aspects of culture are handled also by Ali and Brooks [21, 20].

Hofstede and his followers [667, 670, 868] characterise users based on parameters such as power distance (PDI), individualism / collectivism (IDV), masculinity / femininity (MAS), uncertainty avoidance (UIA), long-term / short term orientation (LTO), and indulgence / restraint (IVR). The adaptation possibilities discussed in [670] are: (1) information density, (2) navigation, (3) accessibility of functions, (4) guidance, (5) structure, (6) colourfulness, (7) saturation, and (8) support. Information density is a special property of content; functionality must reflect (2), (3) (4), and (8); presentation also considers (5), (6), and (7). Edward Hall introduced a classification of cultures based on communication style [298] (see also [484, 614, 752] for [Tables 10.2](#) and [10.3](#) and [Figure 10.5](#)).

Adaptation of WIS to the user culture has recently got more attention in WIS research. Implementing adaptability in IS has several alternative approaches. The most common approach is based on the introduced concept of user modelling, which has its roots in human-computer research. K. Reinecke and A. Bernstein [670] developed an approach to automatic and dynamic adaptation of user interfaces in dependence on the original and current culture. M. D. Myers and F.B. Tan [599] discuss the role of national cultures in IS research.

Pencarelli and Gabbianelli [644] tackle in their conference paper the intercultural aspects in customer management context in global services. Parumusur [639] handles the problems related to organizational development in

global context. Alkandari has studied the problems related to the management of multicultural software development teams [23]. Refinement of WIS can be based on ASM-like refinement [904]. Refinement of the functions may follow the BIER approach [758]. In this case an adaptation can be understood as a refinement [109, 701] of the system, i.e., the new system behaves on the specific restricted scope in the same way as the old system within the same scope.

Personality pattern and screenography approaches have been used for the derivation of user interface styles [360] and for specialisation of principles of visual communication, visual cognition and visual design. We may use stereotypes of users that can be combined into personae [588, 730, 735]. Typical culture sensitive stereotypes are related in communication, leadership, management (organising and division of the work), trust creation between collaborating parties, understanding the motivation factors of individuals and managing the competence differences between them. Stereotype analysis is used by Statkaityte [805, 806] in analyzing the cultural sensitivity of software processes (CSAM model and analysis tool). [393] has applied the stereotype analysis in connection with ITIL service processes.

Key Messages

Cultural Differences

- have a tremendous impact on the acceptance of a web information system by its users;
- can be captured by different cultural stereotypes.

Cultural Awareness in Web Information Systems

- requires presentation guidelines associated with personae, user profiles and portfolios, and life cases;
- requires guidelines addressing the quality of user interaction, development of interfaces, and the handling of adaptation requests;
- can be achieved by extending the storyboard by parametric mini-stories, parametric content schemata, and parametric information search.

Part IV

Rationale of the Co-Design Methodology and Systematic Development of Web Information Systems



The Co-Design Methodology

This chapter is dedicated to a general introduction to the co-design methodology for the design and development of web information systems, which will be developed in detail in the remaining chapters of this book. It will introduce the historical and methodological background that underlies our methodology and draw from related work in other areas. As such co-design is not a methodology that has only come up in the context of web information systems, but instead has a longer tradition in other fields of information systems. We will therefore shed first light into the co-design methodology for data-intensive systems in Section 11.1, which will influence our work on the conceptual modelling of web information systems. We proceed by looking into the area of data-centric services in Section 11.2. Co-design for service-oriented systems already brings up the issue of user needs and demands and the problem of anticipation of user behaviour. In Section 11.3 we widen the scope towards distributed systems with an emphasis on collaboration, another issue that pops up in web information systems. On these grounds we then proceed by motivating the co-design approach for web information systems by giving a glimpse of some cornerstones with respect to storyboarding, involvement of natural language dialogues, and web interaction types. This will be done in Section 11.4. Finally, Section 11.5 addresses the transformation of WIS design to an executable collection of web pages, by means of which the methodology will be integrated into technology that is commonly used in this area.

The abstraction layer model emphasises the design of content on global and local level as expressed by a data and a view specification, respectively. Both together have been investigated intensively in the past in database design (see for instance the large bibliography in [823]), and almost all outstanding questions have been extensively discussed. In addition, functionality has also been addressed through global function specifications and local dialogue specifications (see for instance the approach to dialogue types in [716]), though dynamic aspects have always suffered from a wallflower existence in database design. Nonetheless, it is justified to consider database design as a primary field, in which *co-design* methods have been successfully used, and which con-

sequently should be integrated into the development of methods for modelling, design and development of web information systems.

Web information systems comprise diverse areas such as e-business, entertainment, infotainment, community and identity web systems that are data and information intensive. They integrate a variety of database, workflow and other processing, communication and presentation systems. Their design and development is based on an integrated development of structuring, functionality, distribution, and interactivity of users.

Structuring of a web information system application is concerned with representing the database structure and the corresponding static integrity constraints.

Functionality of a web information system application is specified on the basis of processes and dynamic integrity constraints.

Distribution of web information system components is specified through explicit specification of services and exchange frames.

Interactivity is provided by the system on the basis of anticipated stories for a number of envisioned actors and is based on interaction objects which are used to deliver the content of the database to users or to receive new content.

Database development has been mainly considered as development of database structuring, whereas functionality and interactivity specification have been neglected limiting the impact of the approach for information systems development, in particular as applications, and the required functionality has become more complex. Functionality specification has often been left to workflow engines; distributed applications are often based on an explicit specification of import/export views; interaction support is often not specified at all, but hidden within the interfaces.

The understanding of the *co-design approach* is to integrate these four aspects smoothly into a uniform and sound methodology capturing syntactic, semantic and also pragmatic elements, because a broad variety of users has to be considered. Functionality can be addressed by thoroughly extending the idea underlying the dialogue types; distribution design can be enhanced by means of services provided and explicit exchange frames; interactivity can be better supported through the story space and the supporting interaction type suite.

Thus, the problem of web information system design can be thus stated as follows:

Design the logical and physical structure of a web information system for a given (class of) database management systems and given (classes of) presentation systems, such that it contains all the information required by the user and required for the efficient behaviour of the whole information system for all users. Furthermore, specify the application processes and the user interaction.

The implicit goals of web information system design are:

- To meet all the information (contextual) requirements of the entire spectrum of users in a given application area;
- To provide a “natural” and easy-to-understand structuring of the information content;
- To preserve the designers’ entire semantic information for a later redesign;
- To achieve all the processing requirements and also a high degree of efficiency in processing;
- To achieve logical independence of query and transaction formulation on this level;
- To provide a family of simple user interfaces that are easy to comprehend.

11.1 Co-Design of Schema-Centric Database Systems: The Local-as-View Approach

Centralised database systems exploit a *local-as-view* approach, i.e., while a database as such is specified by a (global) database schema, (local) views are defined by queries on this schema. This makes use of the provided functionality, which offers functions for creation, retrieval (queries), update and deletion (CRUD). This functionality is grounded in an algebra that uses the structure of the database system, so all these functions are defined by algebraic expressions. Views support users and satisfy their data demand and their activities with the database system.

The entity-relationship (ER) model, introduced by P. P. Chen in 1976 [146], has been established as a main conceptual model for database and information system development in applications. The model conceptualises and graphically represents key structural elements of the relational data model. Despite its extensive use in practice it has well known deficiencies, which motivated a large number of extensions to the model, most of which were proposed in the 1980s and 1990s. Many of these extensions addressed the capture of application semantics by means of integrity constraints. Cardinality constraints tackled in [146, 307, 308, 506, 823] capture the most important generalisation of relational database constraints [822]. These proposals have been evaluated, integrated or explicitly discarded in the intensive research discussion surrounding this area. The semantic foundations proposed in [279, 328, 823] and the various generalisations and extensions of the entity-relationship model have led to the introduction of the higher-order or hierarchical entity-relationship model [823], which integrates most of the extensions and also supports conceptualisation of functionality, distribution [833], and interactivity [727] for information systems. Class diagrams of the UML standard are a special variant of extended entity-relationship models.

The higher-order entity-relationship model (HERM)—to be more precise, the higher-order entity-relationship modelling language—provides a language

for defining the structure (and functionality) of database and information systems. While HERM has already been introduced in Chapter 7, we will concentrate now on a deepened treatment of HERM as a modelling language. We will emphasise structural aspects, which will be developed inductively.

HERM supports basic and complex attributes. Basic attributes are assigned to base data types. Complex attributes can be constructed by applying constructors such as tuple, list or set constructors to attributes that have already been constructed. Entity types conceptualise the structuring of things of reality through attributes. Cluster types generalise types or combine types into singleton types. Relationship types capture associations among types that have already been constructed, thus defining a layered structure. Types may be restricted by integrity constraints and by specification of identification of objects defined for a type. Typical integrity constraints of the extended entity-relationship model are participation, look-across, and general cardinality constraints. Entity, cluster, and relationship classes contain a finite set of objects defined on these types. The types of a HERM schema are typically depicted by a HERM diagram.

The main application area for extended ER models is the conceptualisation of database applications. Database schemata can be translated to relational, XML or other schemata based on transformation profiles that incorporate properties of the target systems. The ER model has had a deep impact on the development of diagramming techniques in the past and still influences extensions of the unified modelling language UML. UML started with binary relationship types with look-across constraints and without relationship type attributes. Class diagrams permit n -ary relationship types with attributes. Cluster types and unary relationship types permit the distinction of generalisation from specialisation.

An Example of a HERM Diagram

Let us consider an example of an infotainment WIS as discussed in Chapter 6. The HERM schema is depicted by the diagram in [Figure 11.1](#).

A central element of this infotainment system is the event database, where events are characterised by their title, description, comments, etc. Their identification must be enhanced by a surrogate key eventID if we are not using very complex identification. Events can be separated into different kinds and have some special additional information. Events are categorised depending on their kind. Organisations promote, organise or market these events. People are involved in the marketing and in the maintenance of the database. Events are organised at certain locations at a certain time. The event database also includes legal restrictions, transportation information, and other data that are injected from other databases.

Attributes are identified by the type that uses the attribute. Therefore, we may neglect the unique name assumption. Attributes are typically structured.

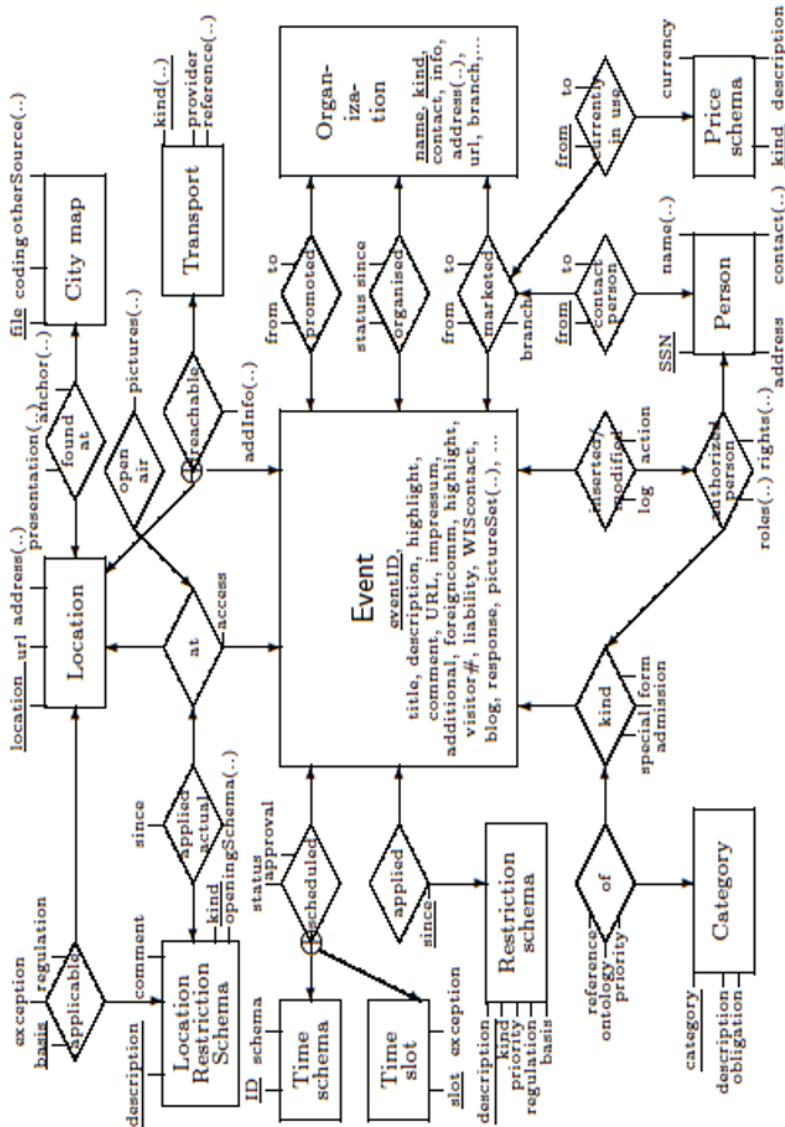


Fig. 11.1. Event database schema in an infotainment application

For instance, the name of a person is given by a sequence of first names, a family name, a set of academic titles, and potentially a family title, e.g.,

name(firstNames <firstName>, famName,
[acadTitles{aTitle}], [familyTitle])

Entity types are represented graphically by rectangles. Attributes primarily identifying a type are underlined. We may use attribute types outside the corresponding rectangle or diamond. Another association is to include attributes inside the type. Relationship types are represented graphically by diamonds and associated by directed arcs to their components. A cluster type is represented by a diamond, is labelled by the disjoint union sign, and has directed arcs from the diamond to its component types. Alternatively, the disjoint union representation \oplus is attached to the relationship type that uses the cluster type. In this case directed arcs associate the \oplus sign with component types. An arc may be annotated with a label.

11.1.1 Static Integrity Constraints

Integrity constraints are used to separate valid states or sequences of states of a database system from invalid ones, i.e., those which are not intended. Constraints are given by users at various levels of abstraction, with a variety of vagueness and intentions behind them and on the basis of different languages. Each structure also contains a set of *implicit model-inherent integrity constraints*:

Component-construction constraints are based on existence, cardinality and inclusion of components. These constraints must be considered in the translation and implication process.

Identification constraints are implicitly used for the set constructor. Each object either does not belong to a set or belongs only once to the set. Sets are based on simple generic functions. However, the identification property may only be representable through automorphism groups [63]. We shall see later that *value representability* or *weak value representability* enable controllable structuring.

Acyclicity and finiteness of structuring supports axiomatisation and definition of the algebra. It must, however, be explicitly specified. Constraints such as cardinality constraints may refer to potentially infinite cycles.

Superficial structuring leads to representation of constraints through structures. In this case, implication of constraints is difficult to characterise.

Implicit model-inherent constraints may give rise to performance and maintenance traps. Often, names or labels are associated with a minimal semantics that can be derived from the meaning of the words used for names or labels. This minimal semantics allows us to derive synonym, homonym, antonym, troponym, hypernym, and holonym associations among the constructs used.

Integrity constraints can be specified based on the B(eeri-)V(ardi)-frame, i.e., by an implication with a formula for premises and a formula for the implication. BV-constraints do not lead to rigid limitation of expressibility. If structuring is hierarchical then BV-constraints can be specified within first-order

predicate logic. We may introduce a variety of different classes of integrity constraints, defined as such:

Equality-generating constraints allow to generate equalities among objects or components of objects for a set of objects from one class or from several classes.

Object-generating constraints require the existence of another object set for a set of objects satisfying the premises.

A class \mathcal{C} of integrity constraints is called *Hilbert-implication-closed* if it can be axiomatised by a finite set of bounded derivation rules and a finite set of axioms. It is well-known that the set of join dependencies is not Hilbert-implication-closed for relational structuring. However, an axiomatisation exists with an unbounded rule, i.e., a rule with potentially infinite premises.

A very important class of integrity constraints of the HERM schema is the class of cardinality constraints. Other classes of importance for the HERM schema are multivalued dependencies, inclusion and exclusion constraints and existence dependencies [822]. Functional dependencies, keys and referential constraints (or key-based inclusion dependencies) can be expressed through cardinality constraints. Multivalued dependencies can directly be represented by ER structures since they separate concerns [840]. Classical cardinality constraints are participation constraints, look-across constraints, and general cardinality constraints.

The diagram in [Figure 11.1](#) can be enhanced by an explicit representation of cardinality and other constraints. If participation constraints $card(R, R') = (m, n)$ are used for a component consisting of one type R' then the arc from R to R' is labelled by (m, n) (see [Figure 11.4](#)). If look-across constraints $look(R, R') = m..n$ are used for binary relationship types then the arc from R to R' is labelled by $m..n$.

11.1.2 Representation Alternatives

The classical approach to database objects is to store an object based on strong typing. Each real life thing is thus represented by a number of objects which are either coupled by the object identifier or supported by specific maintenance procedures. In general, however, we might consider two different approaches to representation of objects:

Class-wise, identification-based representation: Things of reality may be represented by several objects. The *object identifier* (OID) supports identification without representing the complex real-life identification. Objects can be elements of several classes. In the early days of object-orientation it has been assumed that objects belong to one and only one class. This assumption has led to a number of migration problems which have no satisfactory solutions.

Structuring based on extended ER models [823] or object-oriented database systems uses this option. Technology of relational and object-relational database systems is based on this representation alternative.

Object-wise representation: Graph-based models which have been developed in order to simplify the object-oriented approaches [63] display objects by their sub-graphs, i.e., by the set of nodes associated to a certain object and the corresponding edges. This representation corresponds to the representation used in standardisation.

XML is based on object-wise representation. It allows the use of null values without notification. If a value for an object does not exist, is not known, is not applicable or cannot be obtained, etc. the XML schema does not use the tag corresponding to the attribute or the component. Classes are hidden.

Object-wise representation bears the risk of high redundancy, which then must be maintained by the system, thus decreasing performance to a significant extent. Besides the performance problems such systems also suffer from low scalability and insufficient utilisation of resources. The operating of such systems leads to lock avalanches. Any modification of data requires a recursive lock of related objects.

For these reasons, object-wise representation is applicable only under a number of restrictions:

- The application is stable and the data structures and the supporting basic functions necessary for the application are not changed during the lifespan of the system.
- The data set is almost free of updates. Updates, insertions and deletions of data are only allowed in well-defined restricted ‘zones’ of the database.

A typical application area for object-wise storage are archiving systems, information presentation systems, and content management systems. They use an update system underneath. We call such systems *play-out systems*. The data are stored in the same way in which they are transferred to the user. The data modification system has a *play-out generator* that materialises all views necessary for the play-out system. Other applications are main-memory databases without updates, e.g., the SAP database system uses a huge set of related views.

We may use the first representation for our *storage engine* and the second representation for the *input engine* or the *output engine* in data warehouse approaches.

11.1.3 Dynamic Integrity Constraints

Database dynamics is defined on the basis of *transition systems*. A transition system \mathcal{TS} on the schema S is a pair $(\mathcal{S}, \{ \xrightarrow{a} \mid a \in \mathcal{L} \})$, where \mathcal{S} is a non-empty

set of state variables, \mathcal{L} is a non-empty set of labels, and $\xrightarrow{a} \subseteq \mathcal{S} \times (\mathcal{S} \cup \{\infty\})$ for each $a \in \mathcal{L}$.

For such a transition system \mathcal{TS} we introduce a *temporal dynamic database logic* using the quantifiers \forall_f (always in the future), \forall_p (always in the past), \exists_f (sometimes in the future), \exists_p (sometimes in the past).

The most important class of dynamic integrity constraints is *state-transition constraints* $\alpha O \beta$ with a pre-condition α and a post-condition β for each operation O . The state-transition constraint $\alpha O \beta$ can be expressed by the temporal formula $\alpha \xrightarrow{O} \beta$. Each finite set of static integrity constraints can be equivalently expressed by a set of state-transition constraints $\{ \wedge_{\alpha \in \Sigma} \alpha \xrightarrow{O} \wedge_{\alpha \in \Sigma} \alpha \mid O \in \text{Alg}(M) \}$.

Integrity constraints require methods for their enforcement. Integrity enforcement can be realised at the procedural level by application of

- trigger constructs [500] in the active event-condition-action setting,
- greatest consistent specialisations of operations [704],
- or stored procedures, i.e., fully fledged programs considering all possible violations of integrity constraints.

Alternatively, integrity may be enforced at the transaction level by restricting sequences of state changes to those which do not violate integrity constraints, by the DBMS on the basis of declarative specifications depending on the facilities of the DBMS, or at the interface level on the basis of consistent state changing operations.

11.1.4 Specification of Workflows

A large variety of approaches to workflow specification has been proposed in the literature, e.g., in [850] a *basic computation step algebra* has been introduced. Alternatively, the business process modelling and notation (BPMN) language [912] can be used. This language separates users and their roles, e.g., the editor pool with the separation into swimlanes such as the *release session* and the *request to edit* in [Figure 11.2](#). The rigorous specification of BPMN in [443, 444] describes the syntactical and graphical elements as they are used by business analysts, and operators to define and control the business activities (operations on data) and their (event or process driven and possibly resource dependent) execution order. In BPMN a business process is represented by a diagram or graph where nodes are executed and where arcs are used to contain and pass the control or execution order information. The activities are performed in a certain order, depending on resources being available, data or control conditions being true, and events happening. Main elements of diagrams are events (depicted by circles), activities (rounded rectangles), and gates (diamonds) for exclusive, conjunctive, disjunctive or event-controlled split or join. Business processes representing roles may exchange messages (dotted arrow) with other roles.

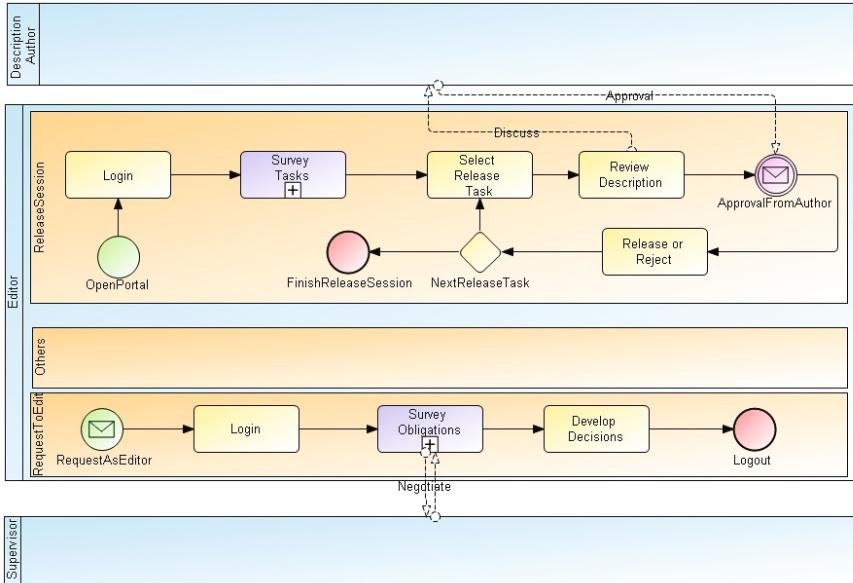


Fig. 11.2. A BPMN diagram for the editing process of an infotainment website

BPMN is a language that supports business process developers in a positive way, yet hinders them at the same time—the “principle of linguistic relativity” [914] postulates that actors skilled in a language may not have a (deep) understanding of some concepts of other languages.

Processes are built by separation of the specification into workflow process and workflow process instances. A singleton process instance is bound to its control token. Inter-process collaboration is supported exclusively through messages and events. Resource dependence is hidden. Swimlanes correspond to different roles of users. Pools are used for views on process sets. Nodes in a diagram are separated into activity, event and control flow (called gates) nodes. Events are either boundary (start, end) or intermediate events. Tasks comprehend only some of possible executions

(Service, User, Receive, Send, Script, Manual, Reference, None).

The rigid localisation in diagrams imposes context-sensitivity of functions, avalanches of side constraints, none-incremental semantics (e.g., goto jumps) and resulting orchestration problems.

11.1.5 View Towers for Information Systems

Information system design methods are often restricted to a three-level architecture, whereas in practical applications hierarchies of views, i.e., views that

are defined on top of views on top of ... on top of schema types. Therefore, *view towers* are already a common implicit folklore background. The research literature does not yet support such view towers. We do not know any publication on view sets after the systematic treatment of relational database development in [235]. Based on our projects we develop in the sequel a novel notion for view towers. This notion extends the notion of interaction types [723].

Relational database technology uses singleton relational view types. View types can be incrementally built on top of view types. The interrelationship between views that are not based on each other remains to be unclear. It is the task of the programmer to develop a clear picture of views. Views have different functions, e.g., views restricting data management to one specific application case, views for security control, views for update control, and views for easing access to data. It is useful to layer these views in a form displayed in [Figure 11.3](#). View set organisation results in an *architecture of views*.

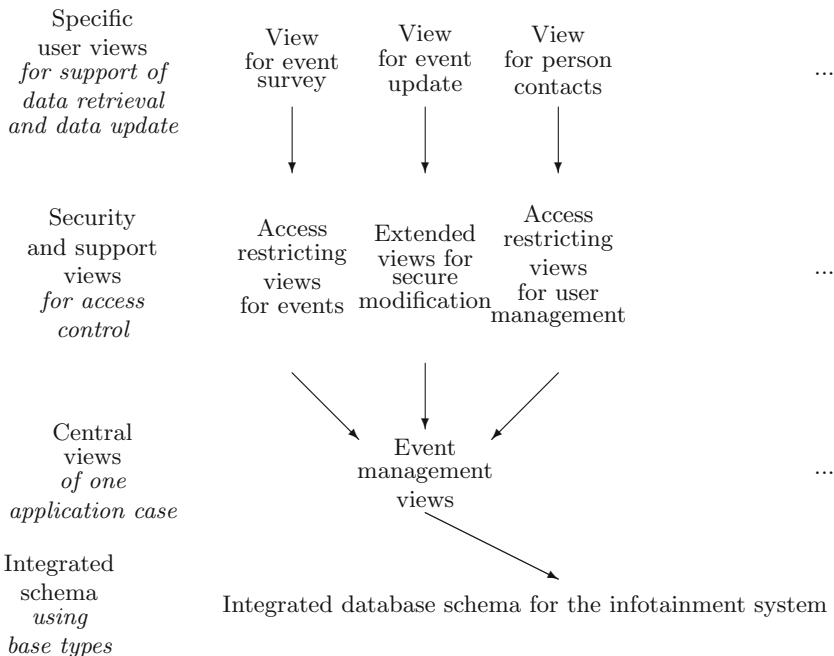


Fig. 11.3. Cutout of a simplified view architecture for event management in an infotainment application

If we wish however to understand, to use, to extend and to maintain such view sets we need a *conceptualisation* of view types. The architecture in [Figure](#)

[11.3](#) is a variant of a *view tower*. This view tower depicts the definitional structure of views, e.g., views for provision of person's contact data are defined on top of views for user management.

HERM Views as Conceptual Views

The HERM schema can be used to define views, e.g., the view schema in [Figure 11.4](#). A singleton view is defined by a query that maps the HERM schema to new types. Combined views also may be considered which consist of singleton views which together form another HERM schema.

A view schema¹ is specified over a HERM schema \mathcal{D} by a schema $\mathcal{V} = \{S_1, \dots, S_m\}$, an auxiliary schema \mathcal{A} and a (complex) query $q : \mathcal{D} \times \mathcal{A} \rightarrow \mathcal{V}$ defined on \mathcal{D} and \mathcal{A} . Given a database \mathcal{D}^C and the auxiliary database \mathcal{A}^C , the view is defined by $q(\mathcal{D}^C \times \mathcal{A}^C)$.

Views can be inductively defined on the schemata generated so far. Thus *view towers* may be defined. View towers support the reuse of views within other view definitions. Typical views in a view tower are:

1. security views for database protection,
2. viewpoint views, and
3. data manipulation views and data retrieval views.

A view tower is a suite of views that are inductively defined. A first-layer view is defined on the database schema. An $(i+1)$ -layer view is defined on views of lower layers and on the database schema. The view definition is enhanced by a maintenance policy. Views can be materialised and enhanced by functions.

Views can have redundant elements, e.g., the attribute *eventID*. In [Figure 11.4](#) we also use two different keys for the type *Event*. A *view suite* consists of a set of views, an integration or association schema and obligations requiring maintenance of the association. The integration is defined over the view schemata. Obligations are based on the master-slave paradigm, i.e., the state of the view suite classes is changed whenever an appropriate part of the database is changed. Views are used to deliver *content* to the user.

Additionally, views should support services. Views provide their own data and functionality. The `group by`, `having`, and `order by` clauses provide a specific representation of the data generated by the view, i.e., by slice or rotate operations. These operations can be extended by dice, drill-down and roll-up functions [840]. We can use the same approach for an extension by functions that support browsing within the data, search, export, input and marking of data. Furthermore, we can use a workplace support by session functions.

¹ Views are typically defined as a schema. This extension of the relational approach where views are single-table views is necessary for maintenance of coherence within a view.

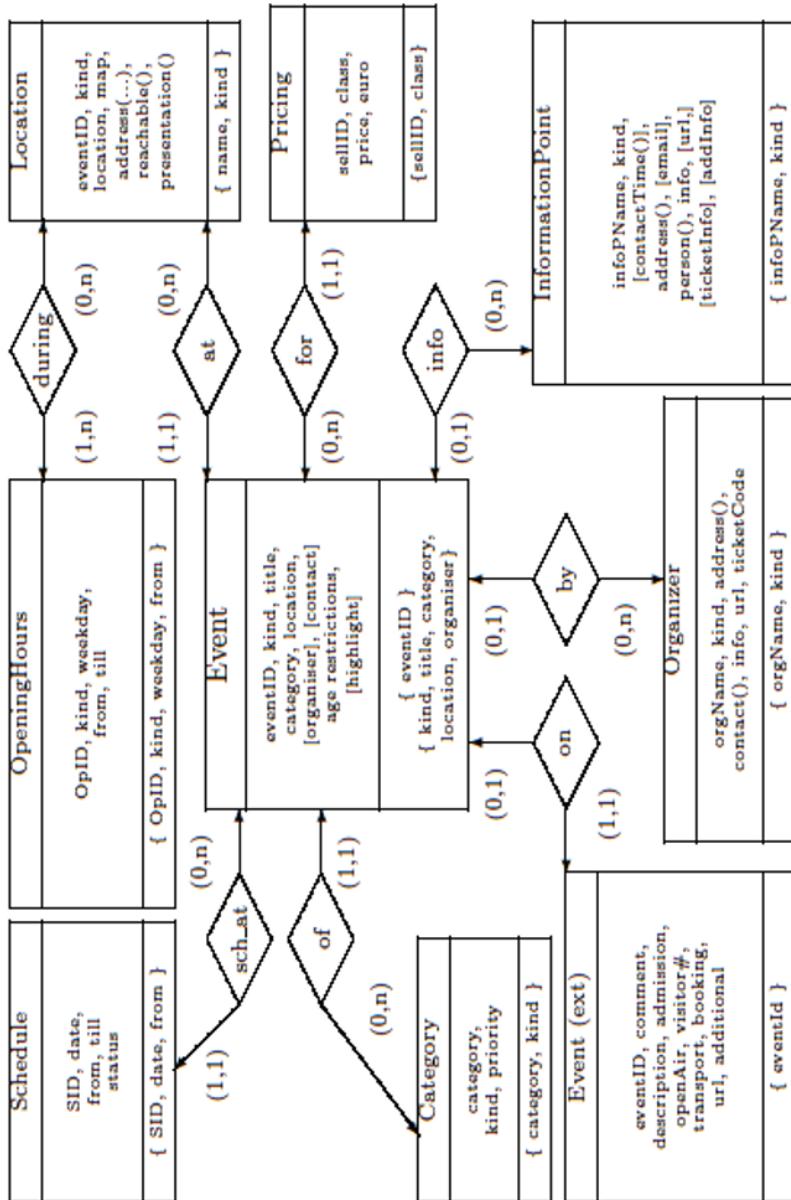


Fig. 11.4. View schema defined on the schema in Figure 11.1

We generalise the view schema by the frame for content and functions:

```

generate MAPPING : VARS → OUTPUT STRUCTURE
  from DATABASE TYPES
  where SELECTION CONDITION
  represent using GENERAL PRESENTATION STYLE
    & ABSTRACTION (GRANULARITY, MEASURE, PRECISION)
    & ORDERS WITHIN THE PRESENTATION & POINTS OF VIEW
    & HIERARCHICAL REPRESENTATIONS & SEPARATION
  browsing definition CONDITION & NAVIGATION
  functions SEARCH FUNCTIONS & EXPORT FUNCTIONS & INPUT FUNCTIONS
    & SESSION FUNCTIONS & MARKING FUNCTIONS

```

These generalised view schemata are the basis for *service specification* [33] (see [Figure 11.6](#)). They can decomposed to relational views and are thus supported by (object-)relational technology. We may develop retrieval views and data maintenance views as well as security views. Additionally, auxiliary views are used for support of insert, delete and update functions for those views that are not directly updateable.

Functions in a view are incrementally defined through expressions in the HERM algebra that incrementally use functions defined for already defined views and types. This specification is now to be extended by the deployment tactics:

```

authorisation USER LIST
  user OBLIGATION
    rights RIGHTS
      with grant OPTION
    roles USER WITH OBLIGATION
  read/update EXCLUSIVITY
enforcement PLACE AND TIME FRAME
  constraint SPECIFICATION
  visibility CONDITIONS FOR STABILITY AFTER UPDATE
    local/global ENFORCEMENT FRAME
  guarantees FOR VALIDITY
materialisation POLICY
  refreshment POLICY

```

The deployment tactic follows the capability relational DBMSs are providing. The view is encapsulated and thus allows access only by authorised users with certain obligations within certain roles based on some transaction approach. View data might obey certain integrity constraints. Despite the view consistency we might also require global consistency to those views and data on which the view is built. View may also be materialised with some policy of direct refreshment or recharge if the underlying data are changed.

A *general view* combines generalised views with some deployment tactics. A *view tower* consists of an incrementally defined set of general views.

Adaptation and Layered Adaptive Systems

Classical information systems development mainly aims at development of the database schema, corresponding views and procedures and functions that might support an application. Adaptation to the user and their needs is sometimes provided by special programs.

Most systems today do not support adaptivity and user orientation. Information as processed by humans is perceived in a very subjective way. As for a knowledge system, the determining factor whether the user can derive advantage from the content delivered is the user's individual situation, i.e., the life case, user model and context. The same category of information can cause various needs in different life cases.

Users typically request or need various content depending on their situation, on material available, on the actual information demand, on data already currently available and on technical equipment and channels on hand. This request is driven by the profile and portfolio of the user, by the national and regional culture, by the organisational and other cultures. Therefore, we need a facility for content and function adaptation depending on the culture and context of the user. Content adaptation and function adaptation may be thus considered as one of the 'grand' challenges of modern internet.

We observed already that culture may be layered into the general system landscape we use, into the national and regional culture and into different organisational cultures. Additionally, personality of a person must be taken into account. We use this layering for stepwise refinement of the view tower in [Figure 11.5](#). Refinement of information systems can be based on ASM-like refinement [904].

Therefore, we develop first an adaptation to the spatial cultures, e.g., the national and regional culture. Next we may adapt the information system to the specific approaches that are used in organisations. We might have to revise the adaptation in the previous step if we have to consider multi-cultural organisations. Finally, we need to adapt the system to the specific personality of the user.

Database operations for view composition are join \bowtie , union \cup , selection σ , special selection or filtering against the data \otimes , projection π , rotation ρ , export cooperation with foreign databases \nearrow , import integration with data delivered from foreign databases \searrow , exclusive choice $+$, and exclusive split Δ .

We base adaptation of a system on rules for transformation of views, functions and workspaces. These rules can be typically applied following the layering in [Figure 11.5](#). They can, however, also be applied in any order. The rule systems for adaptation must obey a Church-Rosser or the confluence property. These properties require that the application of one rule does not hinder the application of another rule. Since such a requirement is not satisfiable we require instead order-independence of a rule system which states that if a rule r_1 is applied and another rule r_2 is not applicable after this application then two other sequences $r_{2,1}; \dots; r_{2,s}$ and $r_{1,1}; \dots; r_{1,t}$ of rules must exist such that

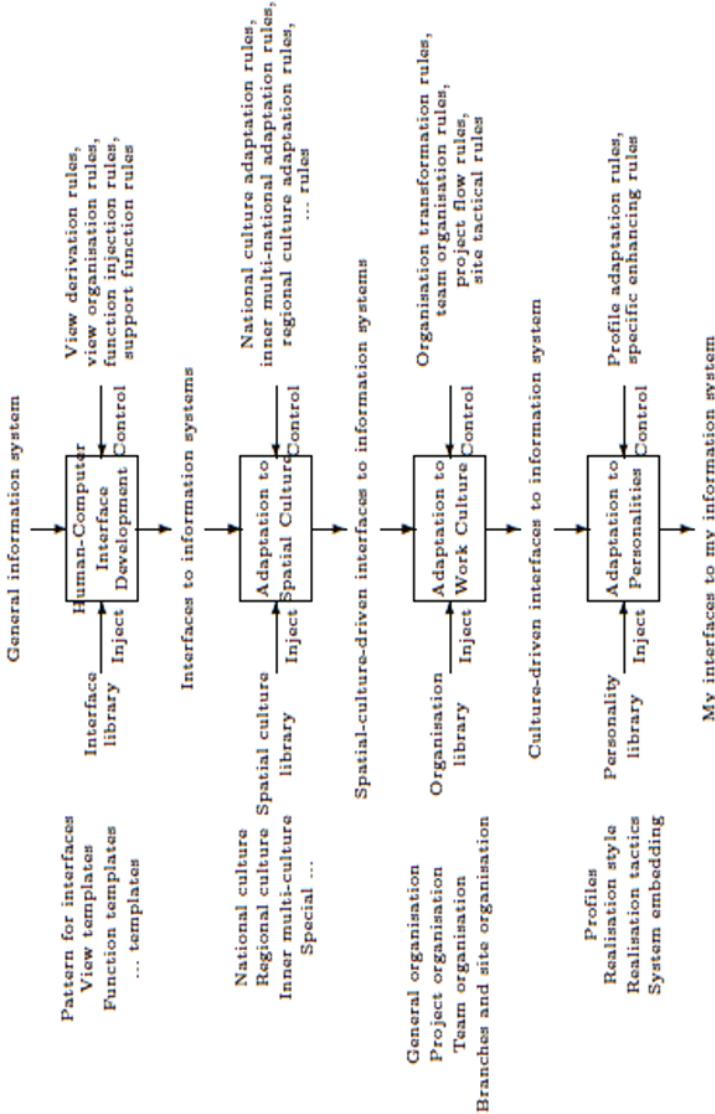


Fig. 11.5. The stepwise adaptation of information systems to human use based on their demands, their spatial culture, their work culture, and their personality

rule application sequences $r_1; r_{2,1}; \dots; r_{2,s}$ and $r_2; r_{1,1}; \dots; r_{1,t}$ have equivalent results.

A formal proof whether a rule system has this property is an open research issue. We can however develop context-free or local rules that have only an

effect on local views, functions or workspaces. It is not yet our goal to develop such confluent rule systems. Instead we show how such a system can be built.

The adaptation of functions and the workplace will not be discussed in this paper. Since the workspace is based on the data and the functions then the adaptation of the workplace has to follow the adaptation of those. The adaptation of the functions may follow the BIER approach [758]. In this case, we should restrict refinement rules to local rules that replace one separable unit by another one. If we restrict our rule system to *unit transforming rules* then we can use graph grammars [207] for adaptation rules.

Our rule system uses the abstract state machine (ASM) approach [110]. It uses the pattern

```
if CONDITION then ACTIONS
```

in which conditions can be event- or data-based as well as control-driven and actions are allowed on the entire system specification as long as this rule application does not conflict with other concurrent application of other rules. In this case an adaptation can be understood as a refinement [109, 701] of the system, i.e., the new system behaves on the specific restricted scope in the same way as the old system within the same scope.

A Small Case Study for a View Tower

Let us consider the view tower from [Figure 10.8](#). Let us restrict the case study to the general view that directly supports the negotiation of field staff agents with the customer. It forms a complex HERM schema. Let us denote the customer view by v_1 , the activity supporting views by v_2 (revival supporting view: $v_{2,1}$, update tracking view: $v_{2,2}$, new customer supporting view: $v_{2,3}$), collaborating companies views by v_3 , agent profile views by v_5 (education view: $v_{5,1}$, work profile view by $v_{5,2}$, responsibility view for field staff by v_{53} .

Additional views provided by the spatial culture, organisation and personality libraries are national and regional customer culture views v_{16} and v_{21} , views for preparation of records or contracts v_{10} and v_{11} , views for contracts with the customer $v_6^{Document}$, an auxiliary view for enabling the contracting with supporting companies v_9 , consultation view with other departments in the company by v_{17} v_{18} ; proposal view v_{25} , and forms view for proposals $v_{5,2}^{Prop}$.

We use adornment indexes for views that are directly defined on its component. The left lower adornments are used for the regional culture, e.g., *BF* and *GB* for the Belgium-Flemish and Germany-Bavarian. The collaboration view with supporting companies typically requires a record of the negotiation policy, e.g., by a view that supports start of negotiations, the negotiation itself and a concluding summary $Start v_{3,1} \bowtie Nego v_{3,2} \bowtie Concl v_{3,3}$. Such negotiation is also followed by an injection of some formal contract extension view $Form v_3$. The right upper adornment is used for shortcuts such as an identifier based compilation of all prerequisites, e.g., v_9^{ID} .

The starting view is the customer view:

$$v_1 \bowtie ((v_{2,1} \cup v_{2,2}) \Delta v_{2,3}) \bowtie v_3 \bowtie (v_{5,1} \cup (v_{5,2} \bowtie v_{5,3}))$$

This view must however be adapted to the national and regional culture, to the specifics of treatment within the given insurance company and to the profile of the agent. Let us consider the case of adaption for the view component v_3 , i.e., for the view that provides data about supporting companies. The view tower for the application can now be composed based on the layering in Figure 11.5 as follows:

- We apply rules that add to customer view data that characterise *prerequisites* (underlined) for involving a supporting company and adapt to the *national culture* (twice underlined):

$$\begin{aligned} v_{16} \bowtie [v_{21} \bowtie] v_1 \bowtie ((v_{2,1} \cup v_{2,2}) \Delta v_{2,3}) \bowtie \underline{v_9} \bowtie v_3 \bowtie \\ (v_{10} \cup v_{11}) \bowtie (v_{5,1} \cup (v_{5,2} \bowtie v_{5,3})). \end{aligned}$$

- Now we can apply rules that allow to consider *regional culture* (underlined), *insurance policy* (twice underlined) and *support proposal* (trice underlined):

$$\begin{aligned} v_{16} \bowtie [v_{21} \bowtie] v_1 \bowtie ((v_{2,1} \cup (\underline{\underline{_{BF}v_{2,2}} + \underline{_{GB}v_{2,2}}})) \Delta v_{2,3}) \bowtie \\ [(\nearrow v_{17} \bowtie v_{18} \searrow)] v_9 \bowtie \underline{\underline{\underline{Startv_{3,1} \bowtie Nego v_{3,2} \bowtie Concl v_{3,3} \bowtie Form v_3}}} \bowtie \\ (v_{10} \cup v_{11}) \bowtie (v_{5,1} \cup (v_{5,2} \bowtie v_{5,3})). \end{aligned}$$

- Since empty parts of the views are not of interest we apply a rule for *filtering* (underlined) the generalised view against the current one:

$$\begin{aligned} v_{16} \bowtie [v_{21} \bowtie] v_1 \bowtie (\underline{\underline{\underline{\otimes_{BF}v_{2,2} \otimes \Delta v_{2,3}}}}) \bowtie [(\nearrow v_{17} \bowtie v_{18} \searrow)] v_9 \bowtie \\ \underline{Startv_{3,1}} \bowtie \underline{Nego v_{3,2}} \bowtie \underline{Concl v_{3,3}} \bowtie \underline{Form v_3} \bowtie \\ (v_{10} \cup v_{11}) \bowtie (\underline{\otimes v_{5,2} \bowtie v_{5,3}}). \end{aligned}$$

- Now a general organisation rule is applied. An insurance contract number covers the *insurance history* (underlined):

$$\begin{aligned} \underline{v_1^{Hist}} \bowtie [(\nearrow v_{17} \bowtie v_{18} \searrow)] \underline{v_9^{ID}} \bowtie \underline{Startv_{3,1}} \bowtie \underline{Nego v_{3,2}} \bowtie \underline{Concl v_{3,3}} \\ \bowtie \underline{Form v_3} \bowtie (v_{10} \cup v_{11}) \bowtie (v_{5,2} \bowtie v_{5,3}). \end{aligned}$$

- Finally we cope with negotiation history, *additional proposals* (underlined) and *forms to fill* (twice underlined):

$$\begin{aligned} \underline{v_1^{Hist}} \bowtie \underline{v_{25}} \bowtie [(\nearrow v_{17} \bowtie v_{18} \searrow)] \underline{v_9^{ID}} \bowtie \underline{Startv_{3,1}} \bowtie \underline{Nego v_{3,2}} \bowtie \\ \underline{Concl v_{3,3}} \bowtie \underline{Form v_3} \bowtie (v_{10} \cup v_{11}) \bowtie (v_{5,2} \bowtie v_{5,3}) \bowtie \underline{\underline{v_6^{Document}}}. \end{aligned}$$

The final view can be easily decomposed into a stream of associated views based on the technology developed for interaction types [723]. The functions and the workplace of the agent are adapted in a similar way.

Advanced HERM Views and OLAP Cubes

The extended entity-relationship model can be used to define an advanced data warehouse model. Classically, the data warehouse model is introduced in an intuitive form by declaring an association or relationship among components of the cube (called dimensions), by declaring attributes (called fact types) together with aggregation functions. Components may be hierarchically structured. In this case, the cube schema can be represented by a *star schema*. Components may be interrelated with each other. In this case the cube schema is represented by a *snowflake schema*. Star and snowflake schemata can be used for computing views on the schemata. View constructors are functions like drill-down, roll-up, slice, dice, and rotate. We demonstrate the power of the extended entity-relationship model by a novel, formal and compact definition of the OLAP cube schema and the corresponding operations.

The data warehouse model is based on *hierarchical data types*. Given an extended base type $B = (\text{Dom}(B), \text{Op}(B), \text{Pred}(B), \mathcal{T})$, we may define a number of equivalence relations eq on $\text{Dom}(B)$. Each of these equivalence relations defines a partition Π_{eq} of the domain into equivalence classes. These equivalence classes c may be named by n_c . Let us denote named partitions by Π^* . The trivial named partition that only relates elements to themselves is denoted by \perp^* . We denote the named partition that consists of $\{\text{Dom}(B)\}$ and a name by \top^* .

Equivalence relations and partitions may be ordered. The *canonical order* of partitions on $\text{DOM}(B)$ relates two partitions Π^*, Π'^* . We define $\Pi^* \preceq \Pi'^*$ iff for all (c, n_c) from Π^* there exists one and only one element $(c', n_{c'}) \in \Pi'^*$ such that $c \subseteq c'$.

We also may consider non-classical orderings such as the *majority order* $\preceq_m^{\text{choice}}$ that relates two named partitions iff for all (c, n_c) from Π^* there exists one and only one element $(c', n_{c'}) \in \Pi'^*$ such that either

$$|c \cap c'| > \max\{|c \cap c''| \mid (c'', n_{c''}) \in \Pi'^*, c'' \neq c'\}$$

or $(c', n_{c'}) \in \Pi'^*$ is determined by a (deterministic) `choice` operator among

$$\{c^+ \in \Pi'^* \mid |c \cap c^+| = \max\{|c \cap c''| \mid (c'', n_{c''}) \in \Pi'^*\}\}.$$

If the last case does not appear then we omit the choice operator in $\preceq_m^{\text{choice}}$.

The *DateTime* type is a typical basic data type. Typical equivalence relations are eq_{hour} and eq_{day} that relate values from $\text{Dom}(\text{DateTime})$ that belong to the same hour or day. The partitions \perp^* , *Days*, *Weeks*, *Months*, *Quarters*, *Years*, and \top^* denote the named partitions of highest granularity, the named partitions of *DateTime* by days, by weeks, by months, by quarters, by years, and the trivial no-granularity named partition, correspondingly. We observe $\perp^* \preceq \Pi^*$ and $\Pi^* \preceq \top^*$ for any named partition in this list. We further notice that *Days* \preceq *Months* \preceq *Quarters* \preceq *Years*, while *Weeks* \preceq_m *Months* is a difficult ordering that causes a lot of confusion.

This notion of hierarchical data types can be extended to complex attribute types, entity types, cluster types and relationship types. These extended types are also called *hierarchical types*. Aggregation functions are defined for extension based data types. The cube definition uses the association between attribute types and aggregation functions

The *grounding schema* of a cube is given by a (cube) relationship type $R = (R_1, \dots, R_n, \{(A_1, q_1, f_1,), \dots, (A_m, q_m, f_m)\})$ with

- hierarchical types R_1, \dots, R_n which form component (or dimension) types,
- (“fact”) attributes A_1, \dots, A_m which are defined over extension based data types and instantiated by singleton-value queries q_1, \dots, q_m and
- aggregation functions f_1, \dots, f_m defined over A_1, \dots, A_m .

A grounding schema is typically defined by a *view* over a database schema. Given a grounding schema $R = (R_1, \dots, R_n, \{(A_1, q_1, f_1,), \dots, (A_m, q_m, f_m)\})$, a class R^C , and partitions Π_i on $DOM(R_i)$ for any component R_1, \dots, R_n , a *cell* of R^C is a non-empty set $\sigma_{R_1 \in c_1, \dots, R_n \in c_n}(R^C)$ for $c_i \in \Pi_i$ and for the selection operation σ_α . Given now partitions Π_1, \dots, Π_n for all component types, a *cube* $\text{cube}^{\Pi_1^*, \dots, \Pi_n^*}(R^C)$ on R^C and on Π_i^* , $1 \leq i \leq n$ consists of the set

$$\{\sigma_{R_1 \in c_1, \dots, R_n \in c_n}(R^C) \neq \emptyset \mid c_1 \in \Pi_1, \dots, c_n \in \Pi_n\}$$

of all cells of R^C for the named partitions Π_i^* , $1 \leq i \leq n$. If $\Pi_i^* = \top^*$ then we may omit the partition Π_i^* .

Therefore, a cube is a special view. We may materialize the view. The view may be used for computations. Then each cell is recorded with its corresponding aggregations for the attributes. For instance,

$$\text{sum}(\pi_{\text{PriceOfGood}}(\sigma_{\text{SellingDate} \in \text{Week}_x}(R^C)))$$

computes the total turnover in week x .

Spreadsheet cubes are defined for sequences $\Pi_1^* \preceq \dots \preceq \Pi_n^*$ of partitions for one or more dimensional components. For instance, the partitions *Days*, *Months*, *Quarters*, *Years* define a spreadsheet cube for components defined over *DateTime*.

The cube can use another representation: instead of using cells as sets we may use the names defining the cells as the cell dimension value. This representation is called a *named cube*.

This definition of the cube can be now easily used for a precise mathematical definition of the main operations for cubes and extended cubes. For instance, given a cube with partitions Π^*, Π'^* for one dimensional component with $\Pi^* \preceq \Pi'^*$, the **drill-down** operation transfers a cube defined on Π'^* to a cube defined on Π^* . **Roll-up** transfers a cube defined on Π to a cube defined on Π'^* . The **slice** operation is nothing else than the object-relational selection operation. The **dice** operation can be defined in two ways: either using the object-relational projection operation or using \top partitions for all

dimensional components that are out of scope. More formally, the following basic OLAP query functions are introduced for a cube $\text{cube}^{\Pi_1^*, \dots, \Pi_n^*}(R^C)$ defined on the cube schema $R = (R_1, \dots, R_n, \{(A_1, q_1, f_1), \dots, (A_m, q_m, f_m)\})$, a dimension i , and partitions $\Pi_i^* \preceq \Pi_i'^* \preceq \top_i^*$:

Basic drill-down functions map the cube $\text{cube}^{\Pi_1^*, \dots, \Pi_i'^*, \dots, \Pi_n^*}(R^C)$ to the cube $\text{cube}^{\Pi_1^*, \dots, \Pi_i^*, \dots, \Pi_n^*}(R^C)$.

Basic roll-up functions map the cube $\text{cube}^{\Pi_1^*, \dots, \Pi_i^*, \dots, \Pi_n^*}(R^C)$ to the cube $\text{cube}^{\Pi_1^*, \dots, \Pi_i'^*, \dots, \Pi_n^*}(R^C)$. Roll-up functions are the inverse of drill-down functions.

Basic slice functions are similar to selection of tuples within a set. The cube $\text{cube}^{\Pi_1^*, \dots, \Pi_n^*}(R^C)$ is mapped to the cube $\sigma_\alpha(\text{cube}^{\Pi_1^*, \dots, \Pi_n^*}(R^C))$. The slice function can also be defined through cells. Let $\text{dimension}(\alpha)$ be the set of all dimensions that are restricted by α . Let further

$$\sigma_\alpha^\sqcap(c_i) = \begin{cases} \emptyset & \text{if } R_i \in \text{dimension}(\alpha) \wedge \sigma_\alpha(c_i) \neq c_i \\ c_i & \text{otherwise} \end{cases}$$

- *Close* slice functions restrict the cube cells to those that entirely fulfill the selection criterion α , i.e.,

$$\{\sigma_{R_1 \in \sigma_\alpha^\sqcap(c_1), \dots, R_n \in \sigma_\alpha^\sqcap(c_n)}(R^C) \neq \emptyset \mid c_1 \in \Pi_1, \dots, c_n \in \Pi_n\}.$$

- *Liberal* slice functions restrict the cells to those that partially fulfill the selection criterion α , i.e., to cells

$$\{\sigma_{R_1 \in \sigma_\alpha(c_1), \dots, R_n \in \sigma_\alpha(c_n)}(R^C) \neq \emptyset \mid c_1 \in \Pi_1, \dots, c_n \in \Pi_n\}.$$

- Lazy and eager slice functions apply the selection functions directly to values in the cells.

Basic dice functions are similar to projection in the first-order query algebra. They map $\text{cube}^{\Pi_1^*, \dots, \Pi_i^*, \dots, \Pi_n^*}(R^C)$ to the cube $\text{cube}^{\Pi_1^*, \dots, \top_i^*, \dots, \Pi_n^*}(R^C)$. Basic dice functions are defined as special roll-up functions. We also may omit the dimension i . In this case we lose the information on this dimension.

Generalizing the first-order query algebra, [823] defines additional OLAP operations such as

- *join functions* for mergers of cubes,
- *union functions* for union of two or more cubes of identical type,
- *rotation or pivoting functions* for rearrangement of the order of dimensions, and
- *renaming functions* for renaming of dimensions.

Our new definition of the cube allows to generalize a large body of knowledge obtained for object-relational databases to cubes. The *integration* of cubes can be defined in a similar form [575].

11.2 Co-Design of Socio-Technical Systems: Database Services in the Task-Centred Approach

Design of information systems is often system-centric. Such systems tend to be unserviceable, non-parsimonious, overly complex, and require learning efforts from the user. Instead, a user seeks information. The notion of information is often given in a syntactic (e.g., entropy-based), semantic (e.g., non-derivable data) or business-oriented (e.g., useful data) form. However, we prefer the anthropomorphic (or pragmatic) notion of the concept of information:

Information as processed by humans, is data perceived or noticed, selected and organised by its receiver, because of his subjective human interests, originating from his instincts, feelings, experience, intuition, common sense, values, beliefs, personal knowledge, or wisdom simultaneously processed by his cognitive and mental processes, and seamlessly integrated in his recallable knowledge.

An information system is thus based on a database system that provides the data to the user. These data are views on the databases. This notion makes it possible to specify the information demand of a user. The *information demand* of a user depends on the tasks the user has to perform and on the skills and abilities of users, i.e., user portfolio and user profile. The user thus demands data in the right form, the right format, the right size and structuring, at the right moment and in dependence on user's tasks and circumstances. The *activity demand* of a user also depends on the portfolio of a user, i.e., on a space of tasks the user has to perform or may perform, on the corresponding obligations and permissions, and on the roles the user is playing in his or her world.

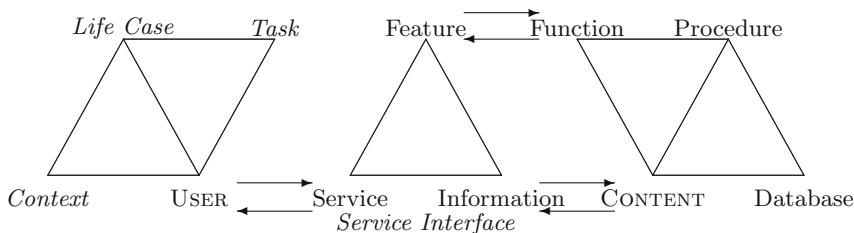


Fig. 11.6. Service architectures for web information systems

Figure 11.6 displays the ‘Janus’ head of socio-technical systems. The user world is driven by life cases, tasks and context. The information system world is composed of a database system, on views defined on top of the database, on procedures which are supported by the database management system, and on functions which support the user work. The service interface is the mediating connector that allows the user to satisfy his or her information and activity

demand. This demand depends on the support needed, e.g., for workplace and workspace requested, for data consumed or produced by the user, and for the environment and context of the user. The user is characterised by a profile, e.g., the work profile, the education profile, and the personality profile.

The mediating service can be supported by *interaction objects* which are specified through *interaction types* [723]. They provide the content and functions in such a form that information and features of a service can directly be associated with the content and the functions.

The Dichotomy of Systems

WIS have two different faces: the system perspective and the user perspective. These perspectives are tightly related to each other. We consider the presentation system as an integral part of WIS. It satisfies all user requirements. It is based on real life cases. The dichotomy is displayed in [Figure 11.7](#) where the right side represents the system perspective and the left side of the ladder represents the user perspective.

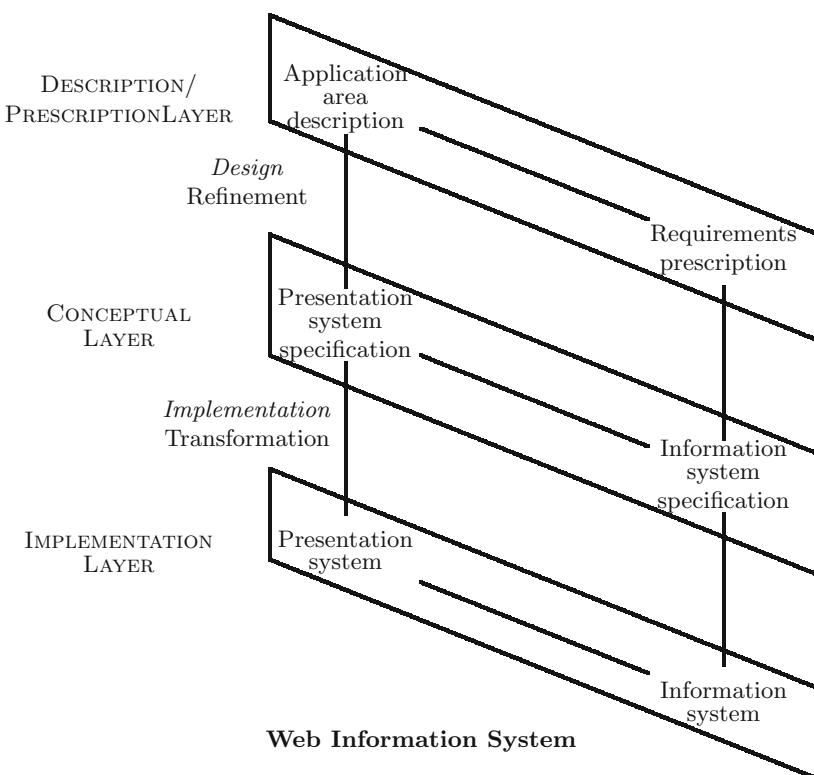


Fig. 11.7. The dichotomy of human-computer systems and the system ladder

Software engineering has divided properties into functional and non-functional properties, restrictions and pseudo-properties. This separation can be understood as a separation into essential properties and non-essential ones. If we consider the dichotomy of a WIS then this separation leads to a far more natural separation into information system requirements and presentation systems requirements. The system perspective considers properties such as performance, efficiency, maintainability, portability, and other classical functional requirements. Typical presentation system requirements are usability, reliability, and requirements oriented to high quality in use, e.g., effectiveness, productivity, safety, privacy, and satisfaction. Safety and security are also considered to be restrictions since they specify undesired behaviour of systems. Pseudo-properties are concerned with technological decisions such as language, middleware, operating system or are imposed by the user environment, the channel to be used, or the variety of client systems.

Properties are often difficult to specify and to check. We should concentrate on those and only those properties that can be shown to hold for the desired system. Since we are interested in proofing or checking the adherence of the system to the properties we need to define properties in such a way that tests or proofs can be formulated. They need to be adequate, i.e., cover what business users expect. At the same time, they need to be implementable. We also must be sure that they can be verified and validated.

The System Ladder

Top-down development of systems seems to be the most appropriate whenever a system is developed from scratch or a system is extended. For this reason, we may differentiate among three layers: the systems description and prescription layer, the conceptual specification layer, and the systems layer. These layers may be extended by the application domain layer that describes the general intention of the system, by the business user layer that describes how business users will see the system and by the logical layer that relates the conceptual layer to the systems layer by using the systems languages for programming and specification. [Figure 11.7](#) relates the three layers of systems development.

The system ladder distinguishes at least between the following refinement layers: description/prescription, specification, and implementation. The refinement layers allow to concentrate on different aspects of concern. At the same time, refinement is based on refinement decisions which should be explicitly recorded. The implementation is the basis for the usage. The dichotomy distinguishes between the user world and the system world. They are related to each other through user interfaces. So, we can base WIS engineering on either the user world description, the systems prescription, the developers' presentation specification, the developers' systems specification. At the later deployment phase we distinguish between the user behaviour and the systems

behaviour. The explicit distinction and the thoughtful integration and refinement of the different perspectives support a sophisticated WIS engineering.

We may extend the ladder by introduction layer, the deployment layer, and the maintenance layer. Since the last layers are often considered to be orthogonal to each other and we are mainly discussing WIS engineering the three layers are out of our scope.

11.2.1 Concerns for Web Information Systems

Modern web information systems reflect at least six different concerns: intention, usage, content, functionality, context, and presentation.

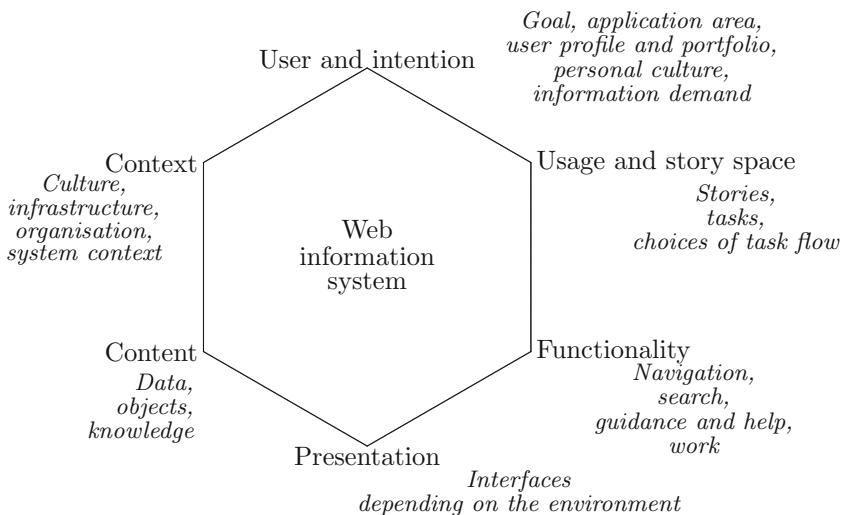


Fig. 11.8. The six concerns for web information system and the conceptual facilities

Intention: The intention aspect is a very general one centred around a mission statement for the system. The primary question is: what is the purpose of the system? Which user shall use the system? Which skills and capabilities can be expected? What kind of support is needed? Users have typically their specific behaviour.

Usage and the resulting story space: Once some clarity with respect to the intentions of the web-based system has been obtained, the question arises how the system will be used by whom. As web-based systems are open systems, it is important to anticipate the behaviour of the users. We model usage through storyboards that allow the specification of the stories users might use with the web information system. The story space is the combination of all stories supported.

Content: The content aspect concerns the question: Which information should be provided to which user at which stage of work? It is coupled with the problem of designing an adequate database. However, the organisation of data that is presented to the user via a website is significantly different from the organisation of data in a database.

Functionality: The functionality aspect is coupled with the question of whether the site should be passive or active. A passive site only allows a user to navigate through the pages without any activity or to search for corresponding content. It can be combined with guidance and help support. In an active site, input is also required from the user. Specific functions allow the processing of user input and the provision of features such as searching, printing, marking, and extraction.

Context: The context aspect deals with the context of the web information system with respect to society, time, expected users, the history of utilisation and the paths of these users through the system. One major element of the context is the culture in all its peculiarities, e.g., national and regional, organisational, website provider, and work environment cultures.

Presentation: The presentation aspect concerns the final realisation by web pages. The presentation can be separated into layout and playout. It follows principles of screenography [583] and is based on principles of visuality such as visual communication, visual cognition and visual design [842]. Design also depends on the support of technical end-devices such as computer screens, television, cell phones, etc., and set layout preferences.

The concerns can directly be represented by separate conceptual structures. For instance, the intention dimension can be specified on the basis of life cases [732] and solutions to them. It is typically combined with a user model with a profile that describes the user properties and with a portfolio of tasks that a user has to complete. Users can be stereotyped. The stereotype must then reflect the specific personal culture of a group of users. [Figure 11.8](#) displays the concerns with their specific elements.

This separation into the six concerns also provides a better understanding of the differences between Web 1.0, Web 2.0, and Web 3.0. Web 1.0 is author driven and uses as stories

- at the provider side `publish/provide_story/support` or `advertise/wait/attract/react/retain` and
- at the user side `inform/subscribe/obtain/answer/come_back`.

Web 1.0 has mainly been oriented towards content provision, which basically meant to deliver content together with a rudimentary functionality. These main functionalities can be:

- navigation facilities for inside site or page navigation;
- acquisition possibilities of information for users from simple content that is based on text, interaction data such as pictures, audio and video data;

- linking facilities;
- search and browse facilities provided to users.

Websites are mainly oriented towards content delivery, provide some functionality and are using a large variety of presentation facilities.

Web 1.0 has made author-driven static content available to numerous users. Users could access exclusively the web pages for researches and personal investigations. The control and management from the ‘top’ did not provide any scope or client-side opportunities for development. This has changed with the evolution of Web 2.0, the so-called social web, as a development process powered by collaborative brainstorming, in which the collective cooperation is at the fore. Meanwhile there are no bounds set to today’s web. With the establishment of user communities, users obtain an abundance of information by high-tech sophisticated services, interchange experiences and benefits by the mass collaboration every single day, because data acquisition and data diffusion are basically accomplished by user interactions inside the whole web story space.

While Web 2.0 integrates collaboration, Web 3.0 provides annotation techniques. These annotation techniques are typically based on linguistic semantics of words used for a reference of data chunks to user semantics. These techniques provide a very good background for sophisticated search and representation techniques. Fully-developed Web 3.0 is characterised by the formula (4C + P + VS) where

- 4C means content, commerce, community and context
- P is used for personalization and
- VS denotes vertical search.

But what is missed in the future of the web, is quality. We want to reach this level of quality with the aid of semantics and pragmatics in respect of the user profile and life cases. We are convinced that lexical semantics composes the base frame of the *Next Generation Knowledge Web*.

[Figures 11.8](#) illustrates the general concerns of websites. It extends the separation of concerns in [Figure 5.1](#).

11.2.2 Application- and User-Driven Design of Systems

Users need information systems to provide a solution for their tasks in dependence on their life cases. Therefore, we characterise the user viewpoint by a profile of the user, by a task portfolio and by their life cases in [Figure 11.6](#). For task completion, users need the right kind of data, at the right time, in the right granularity and format, unabridged, and within the frame agreed upon in advance. Moreover, users are bound by their ability to verbalise and digest data, and their habits, practices, and cultural environment. To avoid intellectual overburdening of users we must first observe real applications before the system development leading to life cases [732]. Life cases specify the

concrete life situation of the user and thus characterise a bundle of tasks that the user should solve. Syntax and semantics of life cases have already been well explored in [729].

User modelling is based on the specification of *user profiles* that address the characterisation of the users, and the specification of *user portfolios* that describe the users' tasks, and their involvement and collaboration on the basis of the mission of the web information system [729].

To characterise the users of a web information system we distinguish between *education*, *work* and *personality* profiles. The education profile contains properties that the users can obtain by education or training. Capabilities and application information which have come about as a result of educational activities are also suitable for this profile. Properties will be assigned to the work profile if they can be associated with task solving information and skills in the application area, i.e., task expertise and experience as well as system experience. Another part of a work profile is the interaction profile of a user, which is determined by the frequency, intensity and style of utilisation of his or her web information system. The personality profile characterises the general properties and preferences of a user. General properties are the status in the enterprise, community, etc., and the psychological and sensory properties like hearing, motoric control, information processing and anxiety.

A *portfolio* is determined by responsibilities and is based on a number of targets. Therefore, the actor portfolio (referring to *actors* as groups of users with similar behaviour) within an application is based on a set of tasks assigned to or intended by an actor and for which he or she has the authority and control, and a description of involvement within the task solution [731]. A *task* as a piece of work is characterised by a problem statement, initial and target states, collaboration and presupposed profiles, auxiliary conditions, and means for task completion. Tasks may consist of subtasks. Moreover, the task execution model defines what, when, how, by whom, and with which data a task can be accomplished. The result of executing a task should present the final state as well as the satisfaction of target conditions.

11.2.3 Services that Satisfy the User Demand

Although services are developed, used, applied, and intensively discussed in modern practice, the concept of an information service has not yet been introduced. Services are artifacts that can be utilised by many users in different contexts at different points of time in different locations and serve a certain purpose. This notion generalises the REA framework (resource-event-agent).

Service systems can be specified through the classical rhetorical frame introduced by Hermagoras of Temnos (Quis, quid, quando, ubi, cur, quem ad modum, quibus adminiculis (W7: Who, what, when, where, why, in what way, by what means)) [174]. Services are primarily characterised by W4: wherfore (end), whereof (source), wherewith (supporting means), and worthiness ((surplus) value). Additionally, the purpose can be characterised by answering the

why, whereto, when, and for which reason W4 questions. The secondary characterisation W14H is given by characterising user or stakeholder (by whom, to whom, whichever), the application domain (wherein, where, for what, wherefrom, whence, what), the solution it provides (how, why, whereto, when, for which reason), and the additional context (whereat, whereabouts, whither, when).

An information service provides information and features. Features allow the user to use the data in the specific form and activity. Typical features are navigation, structuring, query/answer interfaces, export/import support, sessions, and marking. The *service interface* is thus the mediator between the user and the system worlds.

11.2.4 Task-Centred Development for Database Systems as a Service

Information systems support a large variety of users that have different educational, work and personality profiles. These users perform tasks in dependence on their life cases. These tasks can be specified as goal- or purpose-oriented actions. Tasks can be structured into subtasks that are restricted by conditions, data, and context, e.g., organisations, policies, environment, and channel. Tasks form the portfolio of a user.

Task specification is *usage-centred*. Description of tasks includes: identifying essential user and business purposes, understanding targeted users by roles in relation to site, understanding tasks in terms of user intentions and needs, prioritising user roles and user tasks in terms of expected frequency and business importance, and engineering the design of a technical system to fit user and business priorities. Therefore, participatory design relates tasks to services.

Modern information system engineering uses the triptych consisting of the application domain description, the requirements prescriptions, and finally the systems specifications [97, 314]. Main methods for application domain description are life case specification, intention description, profile and portfolio description, and sufficient understanding of context. It describes the application domain as it is or as it should be without any reference to systems. The database system serves as a service in the task-centred approach.

11.2.5 Database and Knowledge Base Systems that Support Services

Database systems that support services support the user by views on the data in the database and by functions. These views deliver information to the user and accept the data of the user. Functions support the features requested by the user. Functions can be complex and are in this case small database programs, i.e., procedures defined in the database management system languages. Views are typically based on queries defined on the database system.

The views and the functions are combined into the interaction object. In a nutshell, an interaction object is defined by an extended view on some underlying database, which can then serve for provision of information and features of an elementary scene. Interaction objects are defined over interaction types and can be combined into containers that deliver or stream them to the user within a web playout system.

The database system may consist of several database systems.

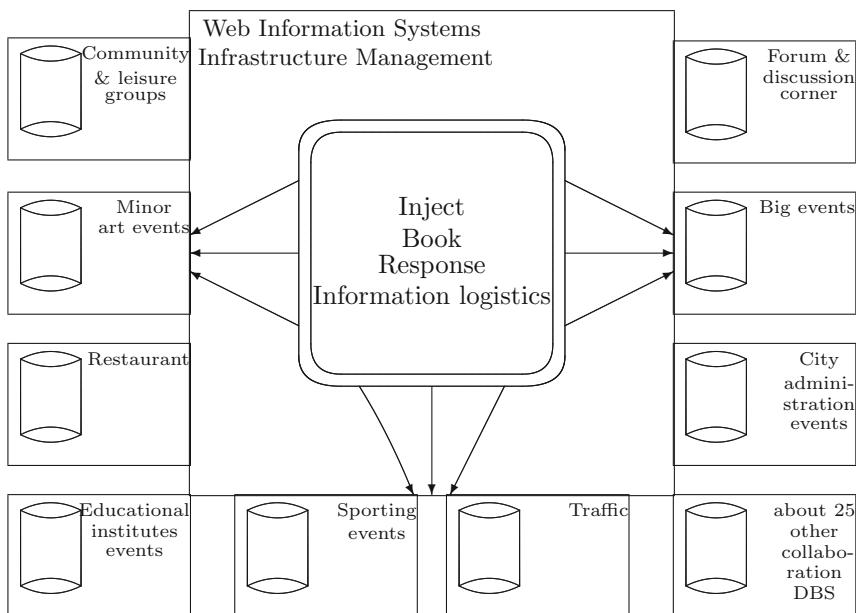


Fig. 11.9. The architecture of a database system providing views and functions for infotainment sites

This combination is necessary whenever a monolithic system cannot be developed. The infotainment system in [Figure 11.9](#) delivers information to visitors, inhabitants and interested users. Infotainment systems are one kind of web information systems. They are more information intensive and mainly support information services. Identity websites are another type of a web information system that is typically less information intensive. Community, edutainment and e-business web information systems satisfy the activity demand of a user as well.

11.3 Co-Design of Distributed Database Systems: The Global-as-View Approach to Collaboration

Specification of distribution has been neglected for a long period of time. Instead of explicit specification of distribution, multi-database systems and federated database systems have been extensively discussed in the literature. On the other hand, database research has succeeded in developing approaches that incorporate conceptual specification and allow to reason on systems at a far higher abstraction level. With the advent of web information systems, systems became naturally distributed.

The information society changed with the advent of the web. Nowadays, applications have become highly distributed. In the past, systems were developed on the whole on the basis of paradigms of programming in the large. This approach still considers systems to be holistic. Monolithic programming is going to be changed to *programming in the web*:

- Flexible, scalable, adaptable communication with variety of protocols and exchange frames;
- Runtime coordination of cooperating partners based on contracts, obligations, permissions and tolerated deviation;
- Exclusive or shared database components;
- Separation agreement for input capsules, output capsules, touchables;
- Exchange architectures for collaboration depending on policy, profile, pattern, style.

11.3.1 Collaboration of Distributed Systems

Distributed systems use services of local databases. These services are defined by views on the local database systems. They provide their own data and functionality. These systems collaborate. Collaboration can be specified in the 3C approach by their three aspects (3C framework):

Communication is defined via exchange of messages and information or classically defined via services and protocols [439]. It depends on the choice of media, transmission modes, meta-information, conversation structure and paths, and on the restriction policy.

Coordination is specified via management of individuals, their activities and resources. It is the dominating perspective of collaboration. The specification is based on the pre-/post-articulation of tasks and on the description management of tasks, objects, and time. Coordination may be based on loosely or tightly integrated activities, may be enabled, forced, or blocked.

Cooperation is the production taking place on a shared space. It can be considered as the workflow or life case perspective. We may use a specification based on storyboard-based interaction [802] that is mapped to

(generic and structured) workflows. The information exchange is based on interaction types for production, manipulation, and organisation of contributions.

The *collaboration* style is based on four components describing:

Supporting programs of the information system including session management, user management, and payment or billing systems;

Data access pattern for data *release* through the net, e.g., broadcast or P2P, for *sharing* of resources either based on transaction, consensus, and recovery models or based on replication with fault management, and for *remote access* including scheduling of access;

The style of collaboration on the basis of peer-to-peer models or component models or push-event models which restrict possible communication; and

The coordination workflows describing the interplay among parties, discourse types, name space mappings, and rules for collaboration.

Collaboration patterns support access and configuration (wrapper facade, component configuration, interceptor, extension interface), event processing (reactor, proactor, asynchronous completion token, accept connector), synchronisation (scoped locking, strategic locking, thread-safe interface, double-checked locking optimisation), and *parallel execution* (active object, monitor object, half-sync/half-async, leader/followers, thread-specific storage):

Proxy collaboration, which uses partial system copies (remote proxy, protection proxy, cache proxy, synchronisation proxy, etc.).

Broker collaboration, which supports coordination of communication either directly, through message passing, based on trading paradigms, by adapter-broker systems, or callback-broker systems.

Master/slave collaboration, which uses tight replication in various application scenarios (fault tolerance, parallel execution, precision improvement; as processes, threads; with(out) coordination).

Client/dispatcher collaboration, based on name spaces and mappings.

Publisher/subscriber collaboration, also known as the observer-dependents paradigm. It may use active subscribers or passive ones. Subscribes have their subscription profile.

Model/view/controller collaboration, similar to the three-layer architecture of database systems. Views (see Subsection 11.1.5) and controllers define the interfaces.

11.3.2 Architectures for Distribution

A number of *architectures* have already been proposed in the past for massively distributed and collaborating systems. In the sequel we use the 3C (or 3K) model for specification of distribution and collaboration. Collaboration will be supported on the basis of exchange frames and information services [512]. The

first specify dissemination, e.g., announcement, volume, time, and protocols. The latter are used for specification of the information service with extraction, transformation, load, and representation. Such distributed services are based on classical communication facilities such as routing, e.g., P2P (as in the case with query based network propagation), such as large nets, and partially closed subnets and propagation.

11.3.3 Coordination Specification and Contracts

Communication and cooperation is nicely supported in the classical setting by communication systems and workflow systems. Coordination specification is, however, still a research problem. Coordination supports the consistency of work products, of work progress, and is supported by an explicitly specified coordinator. If work history is of interest, a version manager is integrated into the exchange support system. The coordination is supported by an infrastructure component. The coordination component observes modification of data that are of common interest to collaborating parties and resolves potential conflicts. The conflict resolution strategy is based on a cooperation contract. The contract is global to all parties and may contain extensions for peer-to-peer collaboration of some of the parties.

The coordinator is based on description of contracts. They describe who collaborates with whom, who supports it, in which scenario or story on which topic, and on which (juridical) basis. Coordination is based on a coordination contract. The contract consists of:

- The coordination party characterisation, their roles, rights and relations;
- The organisation frames of coordination specifying the time and schema, the synchronisation frame, the coordination workflow frame, and the task distribution frame;
- The context of coordination; and
- The quality requirements (ubiquity, security, interpretability, consistency, view consistency, scalability, durability, robustness, performance) for coordination.

We can distinguish four levels of coordination specification. The syntactical level uses an IDL description and may use coordination constructs of programming languages. We use constructs of the JDL (job description language) for this description of resources, obligations, permissions, and restrictions. The behaviour level specifies failure-atomicity-, execution-atomicity- pre-, rely-, guarantee- and post-conditions. The synchronisation level specifies service object synchronisation and paths, and maintains a synchronisation counter or monitor. The fourth level specifies a quality of services level. The coordination profile is specified by a coordination contract, a coordination workspace, synchronisation profile, coordination workflow, and task distribution.

We distinguish between the frame for coordination and the actual coordination. Any actual coordination is an instance of the frame. Additionally,

it uses an infrastructure. The contract specifies the general properties of coordination. Several variants of coordination may be proposed. The formation of a coordination may be based on a specific infrastructure. For instance, the washer may provide a workspace and additional functionality to the collaborating parties.

11.3.4 Exchange Frames for Distribution

Exchange frames may be specified through the triple (Architecture, Collaboration Style, CollaborationPattern).

The exchange architecture usually provides a system architecture integrating the information systems through communication and exchange systems. The collaboration style specifies the supporting programs, the style of cooperation and the coordination facilities. The collaboration pattern specifies the roles of the parties, their responsibilities, their rights and the protocols that they may rely on. Distributed database systems are based on local database systems and follow a certain integration strategy. Integration is based on total integration of the local conceptual schemata into a global distribution schema.

[Figure 11.9](#) illustrates an architecture of a distributed database system. It uses local databases with export facilities as masters in a publish-subscribe collaboration pattern, a master database as the central kernel for information delivery to different users, for customer management, and for integrated information display to different users in an infotainment website.

Beside the classical distributed system we support also other architecture such as *database farms* in [Figure 11.10](#), *incremental information system societies* and *cooperating information systems*. Incremental information system societies are the basis for facility management systems. Simple incremental information systems are data warehouses and content management systems. The exchange architecture may include the workplace of the client describing the parties or parties, groups or organisations, roles and rights of parties within a group, the task portfolio and the organisation of the collaboration, communication, and cooperation.

With the advent of web information systems we face the *heterogeneity trap*: presentation systems of users follow a completely different paradigm and system culture. We thus have to extend architectures, exchange frames and services for such systems.

11.4 The Story Space in the Co-Design Approach

The storyboard approach has already been introduced in Chapter 3. We thus repeat basic concepts and relate them into the co-design approach.

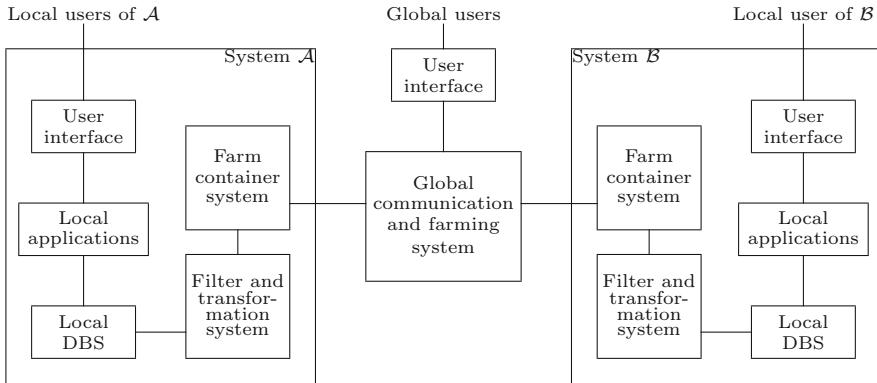


Fig. 11.10. An architecture of database systems farm

11.4.1 The Story Space

The *story space* consists of a well-integrated set of stories and can be modelled by many-dimensional (multi-layered) graphs. A *story* is a run through the story space by a collaboration of users. Typically, each user wants something different; the playout environments can be extremely different; the cultures are distinct; users require a flexible and dynamic adaptation of the system; and they require information and features in a varying granularity. Users may be grouped by the tasks they perform, their profile, and their life cases. Such groups are called *actors*.

A story is composed of scenes. Each scene belongs to a general activity. Scenes combine actions of actors into a *plot*. A scene supports enabled actors. Its usage in a story is controlled by pre-conditions for entering the scene, by accepting conditions for leaving the scene, and by events that open the scene.

Figure 11.11 depicts the general structure of story spaces. Co-design of information systems relates the story space and the scenes to the data and the functions provided by the database system. Additionally, a user produces data that are transferred to the database system. These data and functions are given on the basis of interaction objects [723]. The service interface is typically following specific representation styles. These styles allow to derive the presentation layout and playout (see **Figure 5.1**) in the screenography approach [583]. The layout is based on principles of visual communication, cognition, and design.

The *story space* combines many different stories. The concrete deployment of a web information system by users is a view on this story space, i.e., a *scenario* or path in this space for the given user by abstracting from those scenes for which the user is enabled. Story spaces might be complex for e-business, infotainment (also called information site), edutainment (also

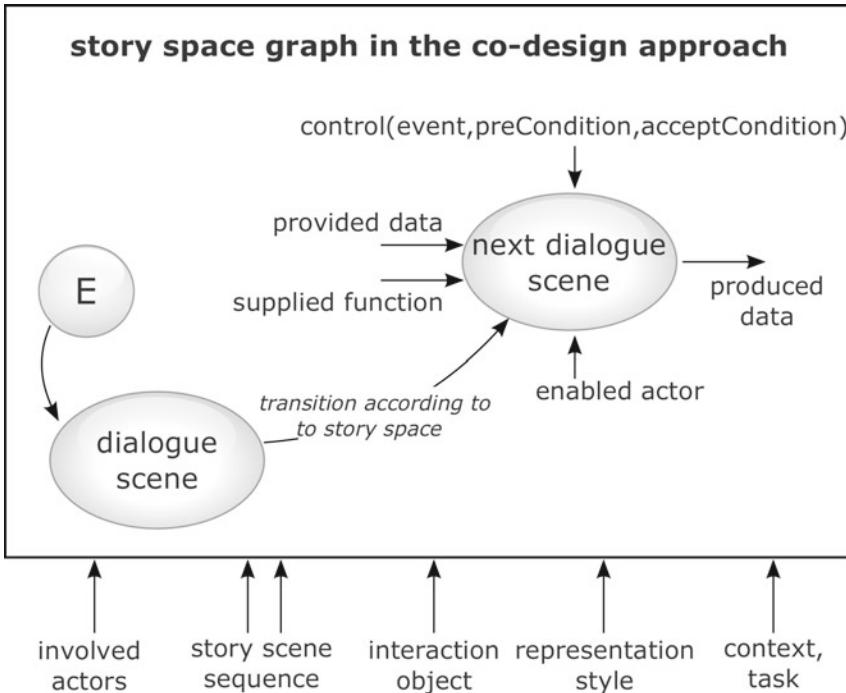


Fig. 11.11. Co-design of stories (through dialogue scene, their control, involved actors, context, tasks, and transitions among scenes), services (information and features), interaction objects (provided and produced data and functions) and guideline for the presentation system

called e-learning site), identity, and collaboration (also called community) web information systems.

For instance, the story space in infotainment contains the editing story. [Figure 11.12](#) combines the stories of the editor, the author of some data, the administrator of the website, and others.

Depending on the role that a user plays, the scenario includes scenes such as content management, release of contributions, or mailbox requests. The workflow in [Figure 11.2](#) is the realisation of such user activities, e.g., in the role of an editor who handles the release of contributions.

11.4.2 Natural Language Dialogues

Chapter 5 introduced the communication act ([Figure 5.2](#)) based on natural language processing. Natural language is the most effective form of human communication. People are used to it and don't need to learn it especially for interacting with a computer as they need it either way. That's why interacting with machines in a natural way is an ambitious and sensible goal. There is

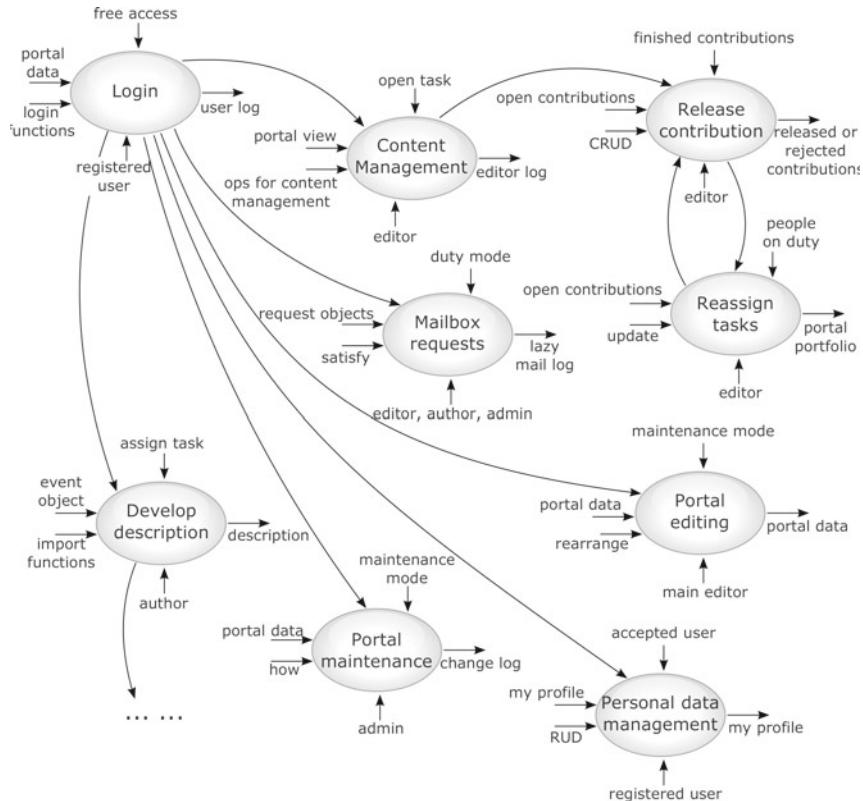


Fig. 11.12. Cutout of the story space for infotainment sites

no need to learn new user interfaces when it is possible to interact with the machine like with man. There are many domains where speech applications increase usability (e.g., navigation systems), productivity (e.g., warehouse-systems) or raise life quality (e.g., for visually handicapped people). But along with all the advantages concerning ease of use there are of course drawbacks. Natural language is an imprecise, fuzzy, interpretable and context-sensitive way of expressing information. Apart from technical context information like physical states in the world model, also the mental model of the user has to be considered. Depending on experience and emotions, different users have a different understanding of statements. We often learn a discrepancy between meaning and saying which often results in the phrase “that’s what I meant”. This gap has to be filled by intelligent questions and contextual analysis.

A dialogue [70, 71] is defined as a conversational exchange of information and interaction between people. It consists of many related utterances with a specific meaning and aim. These utterances are called speech acts. Searle identified five illocutionary acts [768, 769]: assertives, directives, commissives,

expressives and declaratives. Because the process of speaking does not base on abstract mathematical rules, models have to be found to describe a dialogue. To accomplish this, the parts of a dialogue have to be identified. For the exchange of information, outputs and inputs are needed. In natural language it has to be considered that the dialogue flow and the initiative is not fixed. So we have to slightly adopt the model to allow succeeding questions or questions without answers. Imagine an application which allows you on the one hand to inform yourself about touristic events and on the other hand to book these offers. This results in two scenarios: information and booking. The information-scenario consists of several scenes, e.g., determine travel period or get number of persons. These scenes can be divided into different dialogues: *get departure date* or *get arrival date* or *get number of adults* or *get number of children*.

Pre- and postconditions determine if a dialogue can be accessed with the given information and if it has collected/processed the information which is required for proceeding to the next step. This leads to the following syntax:

`on event if precondition do actions accept on postcondition .`

The definition of a scene can be done as follows:

SCENE <i>numberOfPersons</i> InteractionObject: <i>speechForm</i> Actors: <i>customer</i> Context: <i>channel = speech</i> Task: <i>getMandatoryInformation</i> Specification: <code>on travelPeriodGathered if Ø doScene adults = <i>numberOfAd()</i> children = <i>numberOfCh()</i> accept on adults+children > 0 else errordialogue</code> nextScene: <i>accommodationFeatures</i>	SCENE <i>numberOfPersons</i> InteractionObject: <i>submitForm</i> Actors: <i>customer</i> Context: <i>channel = web</i> Task: <i>getMandatoryInformation</i> Specification: <code>on linkClicked if Ø doScene adults = <i>numberOfAd()</i> children = <i>numberOfCh()</i> accept on adults+children > 0 else errordialogue</code> nextScene: <i>accommodationFeatures</i>
--	--

Postconditions can be used to refer to error dialogues depending on the error type. The **nextScene** statement links to the subsequent part of the scene which only is followed if the postcondition holds. The references in **doScene** refer to interaction objects, like the following **numberOfAdults** object, which is the most granular unit in the field of storyboarding.

```
DIALOGUEOBJECT numberOfAdults
EnabledActor: customer
if Ø  
do adults =collect(numberOfAdults)  
accept on adults >= 0 && adults < 10  
else(nomatch) noMatchDialogue  
else errorDialogueNoA
```

These examples are very simple and do not depend on each other. Of course there are situations when succeeding dialog steps refer to previously entered information. Imagine the questions for credit card number and validation code which only make sense when having chosen credit card as payment method. This is realised with preconditions which only activate the scene when certain conditions are true.

11.4.3 Web Interaction Types for Information-Intensive Systems

The classical service-oriented approach in [Figure 11.6](#) is based on a mediator between the user and the technical side. Services satisfy the information and activity demand of users by linking information and features to content and functions. However, if we consider information-intensive services then we extend the technical system with interaction types. In this case, the technical system does not require a mediating system.

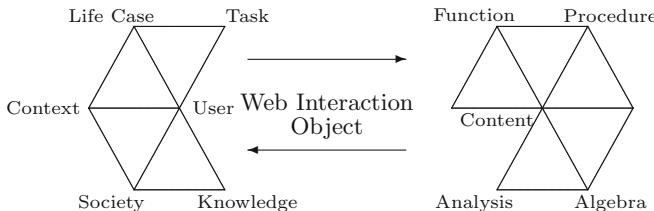


Fig. 11.13. Web interaction objects for information-intensive systems

Instead, a user calls media objects. The architecture displayed in [Figure 11.13](#) neatly supports E-business, edutainment, infotainment, community and identity web information systems based on classical web technologies.

11.4.4 The Onion Approach to Website Realisation

The onion approach [850] to website layout and playout allows the generation of website functions and views for a web information system. On the outer layer, the presentation facilities may be introduced. Typical functions are style and context functions. Containers that contain the interaction objects are mapped to the information and the features that users request. Views may be materialised. If they are materialised then the view handler provides an automatic refreshment support. Thus, we can use the onion system architecture displayed in [Figure 11.14](#).

This layering is nicely supported by XML infrastructures. The database system view data and functions are represented by an XML document suite. The interaction object onion is generated by XSLT rules. Such rules can easily be developed for the containers and finally for the scenes in a scenario.

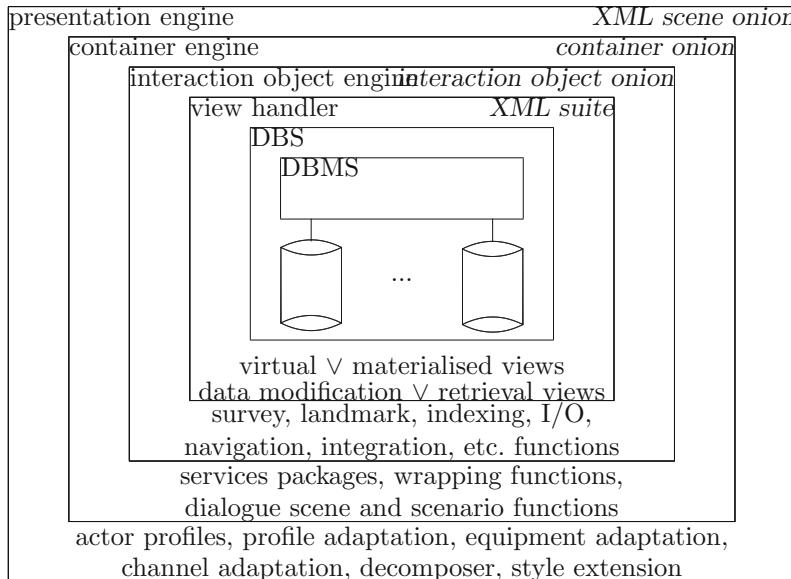


Fig. 11.14. The onion approach to stepwise generation of XML-based sites

Current database management systems such as DB2 neatly support this approach. XML objects are specified by DTD and DAC in DB2. We thus implemented the onion approach based on XML transformations in a form depicted in [Figure 11.15](#).

This transformation approach has already been used in the DaMiT project [742] and implements an XML suite on top of the relational DBMS DB2. The extended ER model [823] provides a better approach to XML suite generation than relational models or the classical ER model for a number of reasons:

- Structures can be defined already in complex nested formats.
- Types of higher order are supported.
- The model uses cardinality constraints with participation semantics.

We observe further that well-structured XML may be considered to be a restricted form of HERM:

- XML Schema and XForms are suited for defining hierarchical extracts of HERM.
- HERM specialisation is based on type specialisation.
- Unary cardinality constraints are supported. If more complex constraints are required we may use vertical decomposition approaches.
- Variants of web objects may be referenced by an annotated XDNL approach.

Therefore, suites of restricted XML documents may be understood as *object-oriented hierarchical database*. If documents are reused by other documents we associate them via XDNL variants.

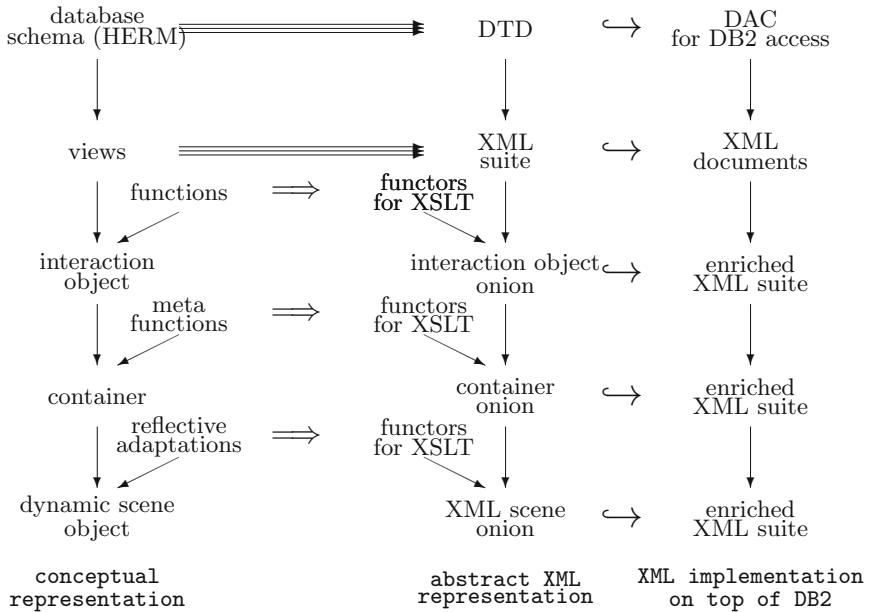


Fig. 11.15. The general procedure for translation from SiteLang to XML

Translation of HERM schemata to DTD can be based on a number of approaches which are similar to the translation approaches used for transformation of schemata to hierarchical database schemata:

Full type separation: each entity, relationship and cluster type is represented by its own `<!ELEMENT ..>` representation. All entity types have an ID which is used through IDREF by other types.

Small star schema representation: the central type of a star schema is represented by its own `<!ELEMENT ..>` representation which uses components for association to other central types. Star associations to other types are represented through attribute lists and the IDREF #REQUIRED data type.

We observe that the translation is not semantics preserving. All referential constraints must be maintained through application programs.

11.5 Transformation of Web Information Systems

The co-design approach is based on the higher-order entity-relationship model. However, most web information systems use XML infrastructures for the lay-

out and playout. Therefore, the transformation of database schemata is enhanced with additional elements of the storyboard and by the distribution specification. XML transformation [422, 823] generalises the transformation of ER schemata to hierarchical database schemata. Additionally, the transformation is based on a selection of XML tactics [424], e.g., Salami slice or Russian doll schema kinds. Furthermore, the `IDRef` attribute value allows the introduction of the concept of the shadow type similar to the CODASYL approach. There is no need to delve into detail here as they are already well-known. There are, however, some specific elements of the transformation process which need a more detailed description.

Our approach to transformation generalises compiler techniques. A typical configuration of a compiler uses *directives*. These directives can directly be mapped to the supporting hardware and software. In the case of web information systems, such directives are patterns for content organisation, content presentation, for navigation, for exploration, and for search and retrieval.

Deriving the Content Organisation Schema

Content must be provided in a form that each user is able to browse or to zap through depending on the information and activity demand. The user profile also contains rules concerning the preferred order of the given user, what content presentation is the most appropriate for what personality, and what data have already been given during the scenario that the user has already completed. Therefore, we add to the content schema ordering decisions depending on ordering criteria, on classification schemata that are available, and on ordering imposed by the story space.

Deriving the Content Presentation Structure

Content presentation is often organised in a linear and sequential form following the reading style of the user community that is under consideration. Users have their personality profile. It allows derivation of the preferences for reading data, e.g., texts. The web supports other presentation structures, such as hierarchical or hypermedia structures. Hierarchical structures are easy to develop and difficult to use whenever data sets become larger. Hypermedia structures are highly flexible. There is, however, a risk of lost orientation. Therefore, they require a more sophisticated browsing, zapping and navigation support.

Content presentation must take into consideration parsimony and other quality of use characteristics. It is a challenging task for a user to scroll through a web page and to remember all of those data which are out of the screen due to scrolling. Therefore, we need to apply techniques that allow the user to see data together with their associated data. The best metaphor for infotainment content presentation seems to be the concertina type cover.

Deriving the Navigation Structure

A website must support users in the space of the website. The simplest technique for orientation in a website is navigation. We distinguish hierarchical navigation based on the tree paradigm, global navigation within a web space based on vertical or horizontal walks in this space, local navigation that extends global navigation in a subspace of the web space, and ad-hoc navigation via meaningful anchors and hyperlinks. Navigation aids are explicit maps throughout the web space, indexing and cataloging schemata, headers and teasers, adaptable guided tours, and meaningful anchors and icons.

Extending Navigation by Exploration Techniques

Users may be supported for navigation by corresponding exploration techniques of the web space. Typical exploration techniques are fish-eye viewing techniques (3D-fish-eye, adorned fish-eye, filtered fish-eye), transformation techniques (radial (locally or globally)), orthogonal (locally or globally), 3-dimensional (implicitly or explicitly), zoom techniques, non-linear techniques (e.g., with the focus point or with multipoint hyperbolic planes) and adoption techniques based on the relevance or importance of the content.

Extending by Retrieval and Search Features

Information intensive WIS must be supported by sophisticated retrieval and search feature for all the seven kinds of search (querying by queries; seeking data by browsing, understanding and refining; property-based questioning; ferreting out data necessary by discovering; searching by associations and drilling down; casting about and digging into the data; zapping through data sets based on search techniques, e.g., uninformed search). If the size of the data is rather small then classical querying approaches do not fit. In this case, retrieval is better provided by exploration techniques. If the data in the databases are not frequently changed and the web information system is traffic intensive then all potential retrieval features may be supplied by pre-prepared materialised or well-indexed data marts.

Additionally, the user language may not match with the interpretation of the WIS, e.g., for quantifiers, connectives and open world questions. Therefore, the semantic space of the user should be mapped to the formal semantic space of the WIS. Furthermore, users need feedback and metadata on the quality of the delivered data. Finally, the presentation of search results can be based on text reading approaches (e.g., Google) or on mindmap techniques (e.g., KartOO).

11.5.1 Mapping of the Website Specification to Business Layer Models

Business use cases [537] are constructed for business processes as bird's-eye views of desired business and system behaviour. They correspond to system features. Business use cases are turned into use cases during requirements analysis. Actors are external, i.e., passive with regard to use cases. However, use cases reflect concerns of system construction, e.g., inheritance of functionality and data ("extends") or composition ("uses").

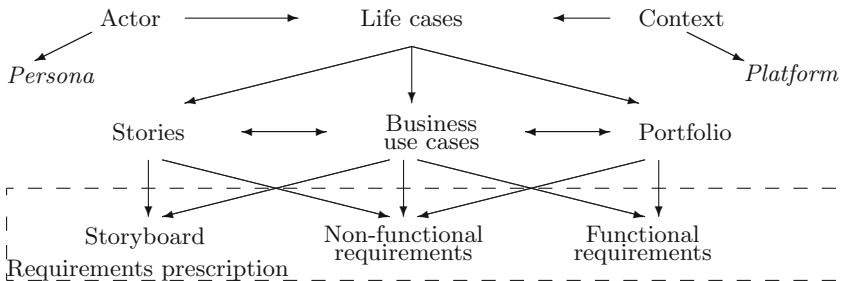


Fig. 11.16. The use of application domain information for requirements elicitation and analysis

We embed our specification approach into classical software engineering approaches that use application domain description, requirements prescription and software specification as three different concerns in the system development triptych [97]. The starting point in our co-design framework are users or a group of users with a similar profile and task portfolio (called actors), their life cases and their context, e.g., the platform to be used. Life cases are the basis for a number of stories, a set of business use cases, and a portfolio that supports all tasks of users.

Classical requirements prescription distinguishes between functional and non-functional requirements. These requirements do not capture the flow of actions that a user performs. Later, business rules and business workflows are added. In most cases they are unrelated to the user stories. They are developed on their own. In our approach, however, we use stories that are already developed and combine these stories into a storyboard. This storyboard can be associated with workflows. A scenario can be supported by several workflows and a workflow can support several scenes of a scenario.

11.5.2 Web Page Extraction

Our approach also allows direct extraction of the web page which supports the basic scenes. Let us consider the login scene in [Figure 11.12](#). Users may play the role of the actors editor, author, main editor or administrator.

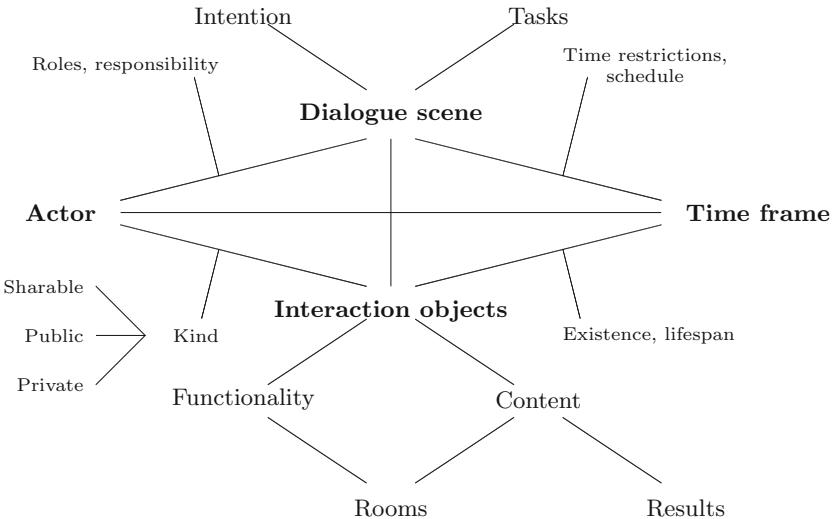


Fig. 11.17. A graphical web page specification frame

A web page carries four aspects: actors, the topic of the page, the interaction objects, and the restrictions, especially the time restrictions. These four aspects are interrelated. Their associations can be visualised in the form given in [Figure 11.17](#). Actors are supported in their work by the interaction objects both for private use (e.g., workspace, unfinished contributions, and their personal workspace) and collaboration use (e.g., shared contributions). One specific interaction object is the personal working room interaction object. It generalises the concept of a session. The work of an actor is governed by the intention and the tasks which must be completed. We combine intention and tasks into the topic of the page. The topic is related to the constraints and restrictions, e.g., time restrictions for the completion of a contribution depending on deadlines and phases within the editing activities. The login web page is supported by interaction objects, e.g., editing interaction objects.

[Figure 11.18](#) shows a diamond representation of the four aspects with their associations in the infotainment example.

11.5.3 Configuration of the Web Page to User Context by Containers

Users, e.g., visitors of a website, may use a large variety of media ranging from small display media such as WAP displays to middle display media such as smartphone interfaces to full screens. We thus need a way to deliver the content and the functions to the user in such a form that the layout is adapted to the current interface.

Most websites that adapt to the user develop different pages for each kind of screen that they support. Since this kind of duplication is laborious, time-

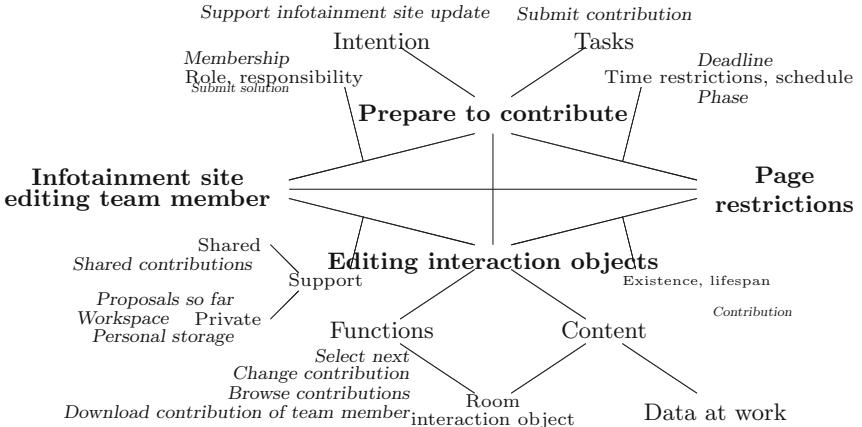


Fig. 11.18. Aspects to be supported by a web page during editing in infotainment

consuming, and error-sensitive, websites have lately been limited to specific browsers and interfaces. However, the theory of interaction types allows the use of a different approach. Each scene is supported by a number of interaction objects. These interaction objects are innerly structured. Therefore, they can be decomposed into several views on top of these objects in dependence on the interface context.

Containers [222] are interaction objects that support content shipping, parametrisation of presentation, and context adaptation. They are used for transfer of information to users according to their current demand and the corresponding dialogue scene. Size and presentation of containers depend on the restrictions of the user's environment. Containers are obtained by adding functions for adaptation and unloading. Scene interaction objects are specialisations of containers by adaption of the container to the user (profile), the environment, and the history. The adaptation of the interaction object to the current context is based on measure rules which support translation of different scales, ordering rules for ordering of components in the container, adhesion rules that maintain coherence of components in an interaction object, and hierarchy meta-rules similar to OLAP functions.

Let us consider the visit of a user interested in sporting events. This activity is supported by the interaction type in Figure 11.4. The adhesion of clubs to events is higher than the one of locations and time to event. Several hierarchies exist such as the time hierarchy (year, month, week, day, daytime) and the location hierarchy (region, town, village, quarter, street), the latter one being fanned. Therefore, the interaction object has the natural internal land map shown in Figure 11.19.

Containers may be extended by dockets which allow tracking of the current usage of the content. They contain data on the content, on the delivery instruction, on the parameters of functions for opening the document, on

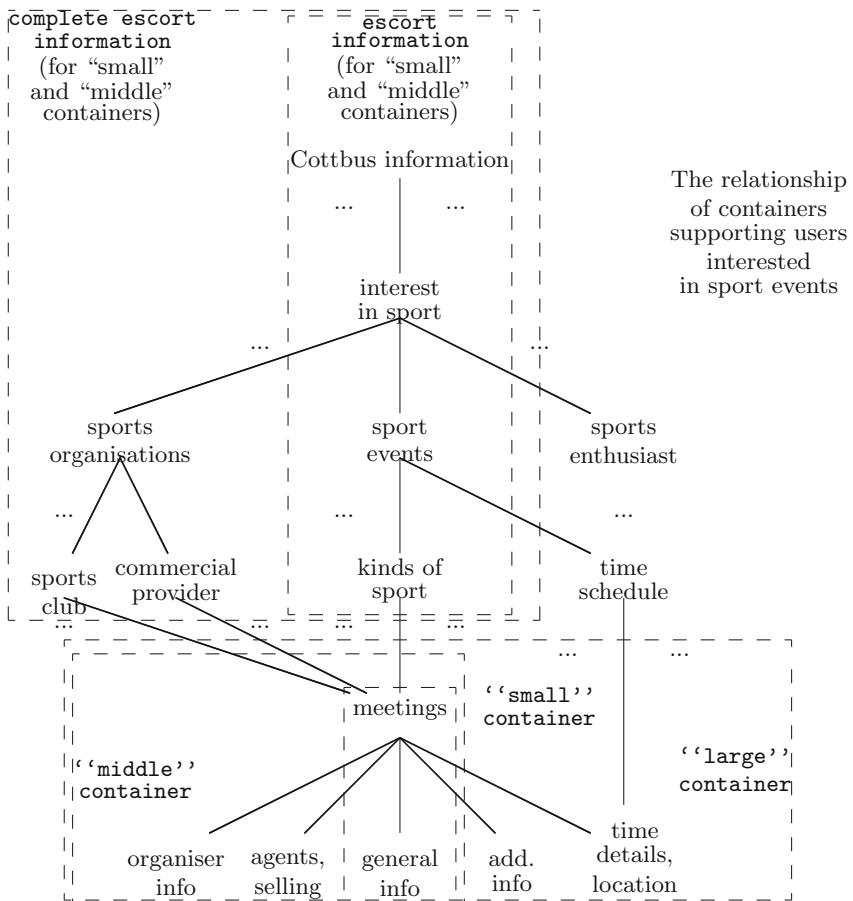


Fig. 11.19. The orientation and information containers satisfying interest in sport events

associations to other interaction objects, on metadata such as resources, restriction, copyright, roles, distribution policy, etc., on the content providers, content reviewers and review evaluators with quality control policies, on applicable workflows and the current status of completion, and on the receipt of the document which enables tracing the interaction object life cycle. The overlay structuring of dockets for containers is supported by the onion approach indicated in [Figure 11.14](#).

Interaction types support a variety of functions such as generalisation, specialisation, reordering, browsing, sequentialisation, linking, survey, searching, and join functions. These functions are also provided by the container.

In [Figure 11.19](#) three different kinds of containers are used: small containers for the most essential and not decomposable components, middle containers with more elaborated data, and full containers for the complete informa-

tion. We can also distinguish an orientation container from the information container. The information container reuses data from the orientation container as escort information.

In the event example, the order is either specified by the scenario in the story space or by the order of information presentation, e.g., it is assumed that information on actual sporting events is shown before information on previous sporting events is given.

11.6 Bibliographical Remarks

Computer science and technology has four sources: mathematics, electronics, engineering and applications [946]. The first two sources are well acknowledged. The scientist adds to the store verified, systematised knowledge of the physical or virtual world. The engineer brings this knowledge to bear on practical problems. Engineering is the art of building with completely different success criteria.²

Engineering is inherently concerned with failures of construction, with incompleteness both in specification and in coverage of the application domain, with compromises for all quality dimensions, and with problems of technologies currently at hand. According to [870], we distinguish eight stages in engineering:

- (1) *Inquire - Discovering the symptoms.*
- (2) *Investigate - Defining the current state.*
- (3) *Vision - Defining the possibilities.*
- (4) *Analyse - Generating a list of potential solutions.*
- (5) *Qualify- Narrowing solutions down to those with the greatest leverage.*
- (6) *Plan - Securing ownership, commitment, permission.*
- (7) *Apply - Managing the realisation of the solution(s).*
- (8) *Report - Measuring the final outcome and capturing experience.*

Methodologies are a body of methods, rules, and postulates employed by a discipline from one side and the analysis of the principles or procedures of inquiry in a particular field from the other side. They are one of the central elements of engineering.

The co-design approach extends finding for classical information systems [823]. This approach has been evaluated in [224, 355] on the basis of SPICE [348] that was developed by international standardization bodies (ISO and

² “Scientists look at things that are and ask ‘why’; engineers dream of things that never were and ask ‘why not.’” (Theodore von Karman; see [695])

“Engineers use materials, whose properties they do not properly understand, to form them into shapes, whose geometries they cannot properly analyse, to resist forces they cannot properly assess, in such a way that the public at large has no reason to suspect the extent of their ignorance.” (John Ure 1998; see [695])

IEC). SPICE is similar to CMMI [155, 155]. It generalises, combines and systematises other approaches such as ARIS [700], the goal-oriented framework KAOS [180], participatory task modelling [630], the Rational Unified Process [465], OOHDMD [683], JustUI [574], HERA [334], and scenario-based programming through life-sequence charts [305].

Methodologies are a research and engineering issue since the development of database management and software systems. We can rely nowadays on settled approaches and well-developed methodologies, e.g., [34, 61, 160, 299, 322, 547, 627, 671, 790, 789, 820]. UML is often used instead of ER, e.g., [34, 590]. A good survey on methodology literature is [176]. Methodologies are supported by patterns [17, 826] that can be understood as parametric schemata and reference schemata [214, 220, 248, 551, 552, 787, 786, 788, 865] that are high-quality schemata and a good start for modelling. These patterns and reference schemata can be constructed using schema algebras for construction [524, 706].

The software crisis has also drawn attention to systematic development of software. A number of frameworks and evaluation approaches have been developed. Similar to SPICE and CMMI assessments [155, 225, 347, 355, 499], we may rate *maturity of a specification* and a modelling approach to:

- (0) A specification is defined in an ad-hoc manner.
- (1) A specification is mainly defined in an informal form.
- (2) Specification development is systematic and managed and the specification is of high quality.
- (3) The specification is based on standards and are well-understood.
- (4) The specification may be optimised, adapted and integrated with other specifications.
- (5) Specification follows a continuously improvable, supporting evolution and migration style.

The maturity ladder and modelling experience we have observed for three decades direct us to the hypothesis that most Computer Science modelling approaches have not yet reached level (2).

The compilation and generalisation of modelling experience is the main issue of design science [28, 377, 469, 550, 895, 919]. Design science – as a problem-solving paradigm for information systems research – seeks to create innovations that define the ideas, practices, technical capabilities, and products through which the analysis, design, implementation, management, and use of information systems can be effectively and efficiently accomplished [59, 197, 318, 550, 896].

Tools for database design and development have been an intensive research issue in the 1980s and 1990s. There are nowadays many tools [510, 924], e.g., DBMain (e.g., [319, 674], ERWin (e.g., [185, 213]), Visual Paradigm (e.g., [876]). Workflow modelling is well supported as well by these tools and many others. Codesign of structuring, workflows and storyboard is supported within the ADOxx tool suite [392, 460].

Views and view ensembles have already been used in [154, 823]. View towers allow to handle viewpoints for a variety of users [360, 361, 362, 697]. Specific conceptual views provide the structural basis for containers [720].

Due to the content challenges in websites and especially infotainment sites, content management systems attracted a lot of research, e.g., [104, 115, 223, 227, 467, 513, 514, 593, 724, 832, 837, 947]. It became the firmware knowledge for website companies, e.g., [7, 162, 754]. Content management is based on the separation of content, concepts, and topics [350, 593, 645, 835]. This separation follows approaches developed in linguistics [165, 166, 309, 310, 441, 937]. Systematic construction of content and concepts can be based on concept nets [83, 72, 404, 388].

Different information system architectures [303, 302, 417, 512] are applicable for WIS. There are many different notions of software and information system architectures [774, 359]. We extend the QuASAR approach to architecture [783] and distinguish seven different views: The conceptual view consists of a rough architecture of the WIS and its components. The application view shows the architecture of the WIS for the business user. The technical view describes main technical components of the WIS, their interaction, their behaviour and their properties. Typically, the technical view is based on layering or multi-tier architectures of services with corresponding exchange frames. The infrastructure view displays the WIS environment, the middleware, the basic software, and the supporting software and hardware. Additionally, incorporation or deployment of components is described. The execution view shows the activity map of the components for different stories. The execution view can be combined with the application view. In this case we use activity zones for depicting behaviour. The module view describes the modules of the system. The program view displays the structure of modules and programs in detail.

User profiles are already important for small WIS [1]. Our approach is based on user profiling developed within the AI community, e.g., [29, 69, 313, 432, 539, 665], and the software engineering community, e.g., [663]. The ARANEUS, HERA, MDA, MDWEnet, OHDM, WebML, WSDM approaches also target on user centred design (see, for instance [47, 69, 144, 334, 577, 640, 685, 761, 803, 183, 182, 888, 894]).

Key Messages

The Co-Design Methodology

- integrates structuring, functionality and distribution and interactivity into a uniform and sound method capturing syntactic, semantic and also pragmatic elements supporting a broad variety of users;

- exploits well developed concepts from conceptual database modelling (local-as-view) to support structuring;
- addresses functionality by thoroughly extending the idea underlying the dialogue types;
- addresses enhanced distribution design by means of service provision and explicit exchange frames (global-as-view);
- addresses interactivity through the story space and the supporting interaction type suite;
- supports an onion approach to stepwise generation of web sites.



Web Information Systems Engineering

The development of web information systems (WISs) is a special and rather specific case for a software development process, and thus WIS engineering (WISE) must be based on a well-defined and manageable methodology. In software engineering the general information systems development process is usually structured into requirements analysis and definition, systems design, systems implementation and testing, and systems operation and maintenance. For web information systems this traditional approach encounters three major obstacles: late integration of architectural decisions, neglection of user expectations, and late implementations. In particular, web information systems must also consider expectations, profiles and portfolios of a large variety of users. In addition, users expect an early involvement in the development and an early evaluation of the system.

However, software engineering research has also produced quality assurance frameworks for the software development process such as CMMI [155, 642] and SPICE [347, 349, 499]. A process in general transforms its inputs to outputs. The software development process is considered to be the set of activities, methods, and practices used in the production and evolution of software [338] and the associated products [642]. The properties of the process have a great influence on its efficiency and on the properties of its outputs. It is widely accepted that the quality of a software product is largely determined by the quality of the process used [940].

The co-design methodology is based on a step-wise refinement along the abstraction layers. Since the four aspects of information systems - structuring, functionality, distribution and interactivity - are interrelated they cannot be developed separately. The presented co-design framework is underpinned by a methodology for stepwise development. It has been assessed and is testified [224] to be on level 3 in the SPICE framework. Maturity of methodologies for systems development can be characterised by six *capability levels* [347]: (0) incomplete, (1) performed, (2) managed, (3) established, (4) predictable and (5) optimizing. The capability is measured based on process attributes. The indicators of process capability are generic practices, generic resources

and generic work products. For example, the process attributes on level 2 are (2.1) performance management and (2.2) work product management, and respectively on level 3 (3.1) process definition and (3.2) process deployment.

Unlike SPICE the structure of SW-CMM (Capability Maturity Model (CMM) for Software; CMM integration (CMMI)) [499] is one-dimensional. SPICE separates processes and capability levels into two dimensions while CMM handles them in one dimension. Also the object to measure is different. CMM focuses on an organization's maturity whereas SPICE focuses on a single process capability. There is also some dissimilarity concerning the terminology. e.g., processes in CMM are called process areas. SW-CMM is composed of five maturity levels (1-5). With the exception of the Initial Level, each maturity level is composed of several key process areas. Each key process area consists of key practices, which are organized into five sections called common features: commitment to perform, ability to perform, activity performed, measurement and analysis, and verifying implementation. In addition to key practices the key process areas are characterized by purpose and goals. The key practices are illustrated by sub-practices and supplementary information. A key practice may refer to key process areas or other key practices.

12.1 The SPICE Methodology to Development of WIS

12.1.1 Application Domain Description, Requirements Prescription and Systems Specification

Bjørner and Heinrich consider software engineering on the basis of the triptych consisting of the application domain description, the requirements prescriptions, and finally the systems specifications [97, 314]. WIS specification is oriented towards systems that are easy and intuitive to use. Therefore the classical information systems development model adds to systems development the development of presentation systems too. Classically user interface are built after the system has been developed. In this case, the user has to learn how the system behaves and must adapt his or her behaviour to the systems behaviour. We repair this mismatch by primarily considering the user worlds, the user stories, and the applications.

The main aim of WIS development is software that can and will be sold. Hence it must be a system whose use pleases people and which solves a problem. Therefore, the specification of the application domain is of primary importance.

Requirements must be reasonably well understood before software can be designed, programmed, and coded. The application domain must be reasonably understood in all its dependencies, weak and sufficient properties before requirements can be expressed properly.

Software engineering classically considers only the second and the third dimension. Application domain engineering is understood as the pragmatical side of the storyboard approach.

The software engineering triptych consists of three areas: description of the application domain, prescription of systems requirements, and specification of software systems. Within our approach we extend these areas by description, prescription, and specification of the presentation system. We may, furthermore, specialise to information systems. Therefore, the general development can be based on the ladder depicted in [Figure 11.7](#).

Application Domain Description

The application domain description is the first dimension of website development. Application domain engineering has not yet got its foundations. Our website development methodology includes this phase as the first phase of the development. Life case specification, intention description, profile and portfolio description, and sufficient understanding of context form the main methods for application domain description. It describes the application domain as it is or as it should be without any reference to systems.

It contains descriptions of the phenomena that can be observed and descriptions of concepts, i.e., abstractions that these phenomena embody

- things,
- functionality,
- activities, and
- concepts.

The application domain is described by a number of properties, phenomena and various aspects that are desirable. A description includes designations, definitions, and refutable assertions. It can be formal or informal, sets a scope and a span, and expresses moods. Designations consist of a name, a recognition rule which purports to designate the phenomenon, and a general specification of the set of possible interpretations. Lexical definitions state the meaning of an expression in terms of other expressions. Ostensive definitions point to examples. Stipulative definitions assign a new meaning to an expression. Definitions may set bounds. Refutable assertions are claims that may be shown to be true or to be wrong.

Application domain description can be based on the following major stages:

1. modelling business processing facets of an application domain,
2. modelling intrinsic facets of an application domain,
3. modelling possible support technology facets of an application domain,
4. modelling possible management and organisation facets of an application domain,
5. modelling possible rules and regulations of an application domain,

6. modelling possible script (story) facets of an application domain, and
7. modelling possible human behavior facets of an application domain.

The life case specification together with the context description, and the user description which is based on the profiles of user combines these major stages. So one can use this description for an elegant and sophisticated elicitation of requirements and for the derivation of the presentation system specification.

Metaphorical structures, i.e., metaphors, allegories, metonymies or synecdoches can be used to help the scanning user understand the WIS. In general, metaphorical structures have a communicative or cognitive function. For example, many users don't realize that within a computer context a trash can often stands as a metaphor for the action of deleting files. The users can understand content, functionality, and intention intuitively via using such metaphorical structures. WIS development can profit from well-integrated metaphorical structures aimed at focus on a deeper context explanation as well as on a more specific navigation and selection. The power of metaphorical structures can be best exploited if they are used for complex actions and in an integrated way together with structure, functionality, and interfaces.

Requirements Capture and Requirements Prescriptions

Requirements are based on a description of properties that a system shall maintain or offer. They are compiled into a documentation which prescribes desired properties of a machine, i.e., what the machine should and should not offer of data, control and functions. Since machines do not operate without an environment, environment description is often included into requirements.

In almost all software engineering texts requirements elicitation and acquisition is considered to be the first phase of systems development. In our methodology it is the second phase. It is already based on an abstract notion of a machine. We use for machines notions developed for abstract state machines (ASM) [110]. ASM support software engineering by a notion of machines that describe the basic virtual machines for any kind of computation, that support general and refined specification of the (basic) programs, that allow to express and to reason on the basic properties of such machines and programs, and that provide means for analysis of machines.

There are four basic kinds of requirements we want to capture:

- Business process re-engineering: reformulation of previously adopted descriptions together with additional activities to be performed and change management to be applied.
- Application domain requirements: These properties are based on application domain phenomena and concepts. They are transformed and refined to properties of the system we need to preserve.

- Interface requirements: They are expressed in terms of such concepts that are shared by the machine and the application domain. Typical shared concepts of the story space and the system space are shared initialisation requirements, shared refreshment requirements, computational data and control requirements, human-computer physiological requirements, and HCI dialogue requirements.
- Machine requirements: These requirement are solely expressed in terms of the machine. We distinguish performance requirements, dependability requirements, maintenance requirements, platform requirements, documentation requirements, and other machine requirements.

System requirements are the union of these system requirements. Requirements are given by a number of observable attributes, impairments or threats causing dissatisfaction, methods for measurement, and means or techniques for satisfaction. Methods for measurement may be based on metrics. This pre-scription framework can be extended to a quality assurance framework on the basis of means.

For instance, accessibility, availability, integrity, reliability, robustness, safety, privacy, and security are attributes for dependability requirements. Procurement (fault prevention or tolerance) and validation (fault removal or forecasting) are typical means. Threads to dependability are faults, errors and failures. Fairness is a typical accessibility attribute. It means that all users will have a guarantee for the access to the machine resources, e.g., computing time or space.

Requirements acquisition is based on the existence of an application domain model. Partners have thus already agreed on the system, on the concepts to be supported, on the functionality required, and on general properties to be satisfied. A number of questions must be answered during requirements acquisition. The selection of the systems architecture is one of the most essential problems. It includes the choice of the platform on which to implement the machine. Other decisions made during requirements acquisition result in business process re-engineering, interface requirements, and machine requirements. Acquisition includes elicitation, structuring, ordering, indexing, prioritisation, and proper recording or documenting.

Elicitation is typically a difficult task. The difficulty is caused by:

- Thin spread of application domain knowledge: It might be distributed across many resources and is rarely available in explicit form. There might be conflicts between different resources and partners due to intentions and portfolio.
- Tacit knowledge: The complete and consistent description of applications becomes harder if the application is in regular use.
- Limited observability: The partners might be too busy or too concentrated while using the existing system. The presence of an observer may change the behaviour.

- Bias: Partners may not be free or may not want to describe what should be described due to the political climate or organisational factors or due to the expected outcome.

A number of elicitation techniques can be applied during WIS engineering.

- Traditional approaches use introspection, analysis of existing documents and data interview techniques, prototyping surveys and questionnaires, and techniques based on group collaboration for elicitation. The last techniques are based on focus groups, brainstorming or extreme development workshops. These techniques are often not applicable, may be extremely misleading or biased, compromised by outdated data or large amount of qualitative data, do not support comparison among different respondents, are difficult to master.
- Representation-based approaches are either intention based, interaction scenario based or use specific representation techniques such as use cases. These techniques are not scaling with the evolution of intentions, use either very complex goal hierarchies leading to analysis paralysis or lack of detail, lack structure since we need stories to provide higher-level view, do not capture domain knowledge and are sometimes confused with a precise specification.
- Contextual (social) approaches use ethnographic techniques such as participant observation and ethnomethodology discourse analysis techniques based on conversation analysis or speech act analysis portfolio-oriented techniques of participatory design, socio-technical methods, or soft systems analysis. These techniques must uncover the order of the social world that is often not immediately obvious or describable, structured or based on pre-given categories. Naive observation may be extremely time consuming.
- Cognitive approaches are based on task analysis and behaviour protocol analysis. The latter technique is essentially based on introspection and is hence unreliable since it has no social dimension.

The life case study combines cognitive techniques with contextual approaches. It is extended during WIS utilisation analysis by traditional approaches. This combination allows to avoid the known disadvantages of the more specific elicitation approaches.

Structuring is a relatively simple task if templates are used for prescription. Ordering is based on the above mentioned template for requirements. We can order requirements based on the naming schema used for naming, by the attributes describing the requirement, by threads they are causing, and by techniques for satisfaction. The same schema can be used for indexing. Since we use blackboard techniques for development in the the co-design approach we prioritise requirements and associate them with steps or processes. The co-design approach is based on a proper recording or documenting of each step.

Requirements prescription results in a IS development and system documentation that covers a number of topics. We use the following template for structuring the IS development and system documentation: prescriptions of stakeholders, activities, and sketches; facets and aspects of the system to be developed; machine properties such as performance, dependability, maintenance, platform and documentation properties; analyses of concepts, their validation and verification with a requirements theory and methods for reasoning on satisfiability and feasibility; terminology.

WIS Specifications

After completing the requirements prescription the WIS specification is going to be developed. It consists of two parts: *Presentation system specification* and the *information system specification*. Media types and containers are the glue between the two parts of the WIS. We thus may consider them as specific middleware types. They support the story space by data transportation and data deployment.

WIS specification is often based on an incremental development of WIS components, their quality control and their immediate deployment when the component is approved. The development method is different from those we have used in the first layers. Application domain description aims in capturing the entire application based on exploration techniques. Requirements prescription is refining the application domain description. Specification is based on incremental development, verification, model checking, and testing. This incremental process leads to different versions of the WIS: demo WIS, skeleton WIS, prototype WIS, and finally approved WIS.

WIS specification is combined with either eager or lazy consecutive quality management. In eager quality management each of the properties required is constantly verified whenever a change in the specification has been made that might have an impact on the property. Lazy quality management either requires the maintenance of a monitor or uses checkpoints for issuing a quality check for all quality criteria. Both quality management approaches use verification, model checking and testing methods.

WIS specification is at the same time a discipline, a science, a craft, a logic, and a practice. We may use a number of specification practices such as structured specification, extreme specification, object-oriented specification, and chief developer specification. Structured specification is based on the notion of modules. These modules form a technical architecture. The execution model of modules is often based on hierarchical depth-first execution. Extreme specification is based on techniques specification in small teams with escorting business user evaluation. Object-oriented specification is based on forming high-level specification objects and then developing them further. Chief-developer specification is based on approval of design decisions by a chief developer.

12.1.2 Dimensions for WIS Engineering

Since 1998 the technical report ISO/IEC TR 15504 [347, 349] has provided a framework for the assessment of software processes. It is generally known as SPICE (Software Process Improvement and Capability dEtermination), a term that originally stands for the initiative to support the development of a standard for software process assessment. The international standard ISO/IEC 15504, with five parts, has been published in 2003-2006.

Process capability is a characterization of the ability of a process to meet current or projected business goals [346]. Part 2 of the standard defines a measurement framework for the assessment of process capability. In the framework, process capability is defined on a six point ordinal scale (0-5) that enables capability to be assessed from Incomplete (0) to Optimizing (5). The scale represents increasing capability of the implemented process. The measure of capability is based upon a set of process attributes, which define particular aspects of process capability [347, 349].

Web information systems engineering models the activities performed or managed by people participating in the engineering process. Models support understanding of the application domain and of the WIS, can be a source of inspirations, can be used for presentation and training, are abstractions that assert and predict behaviour of the WIS, and are the source for the implementation of the web information system.

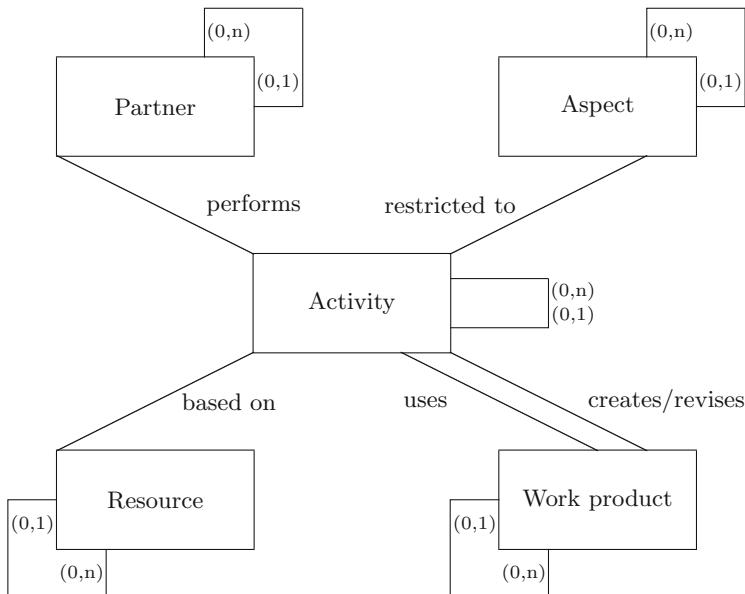


Fig. 12.1. The primary dimensions of information systems engineering

Activities may result in work products, may revise existing work products, or may be based on different work products. Work products considered in our framework are: developed documents and restrictions to the completion of the activity (postcondition). We therefore distinguish between the five primary dimensions of WIS engineering:

Activities (“how”) describe the way how the work is performed and the practices for the work.

Work products (“what”) are the result of the specification and are used during specification.

Roles (“who”) describe obligations and permissions, the involvement of actors in the specification process.

Aspects (“where”) are used for separation of concern during the specification process.

Resources (“on which basis”) are the basis for the specification.

These five main dimensions are hierarchically structured. We structure the associations among the dimensions in [Figure 12.1](#).

These five dimensions must be mapped to technology. This mapping includes the derivation of the general solution and of the software architecture. The five dimensions can be enhanced by secondary dimensions. Finally, each object of one dimension may be associated to other objects of the same or other dimensions. We observe in the resource dimension, that, for instance, methods are founded on theories.

The Activity Dimension

The system development process is usually decomposed into clearly distinguishable development activities that support tackling separate issues at separate times. An activity either starts from scratch and results in a completed work product or starts from a set of work products and results in a completed or revised work product. Activities are going to be based on steps in our framework. We consider activities as work product transformations. Therefore, the steps can be given through refinement activities, i.e., elicitation, determination, instantiation, extension, and fitting.

In this paper we mainly restrict the consideration to engineering activities. These activities are enhanced by acquisition activities and activities supporting the life cycle. The substructure of activities we use is sketched in [Figure 12.2](#).

Each specification methodology possesses a process model, i.e., a model of how the stages (layers) and steps of specification activities follow one another, how they may iterate, how they can be applied in parallel, and what quality or capability properties must be satisfied. The design and development quality depend on four main success factors: structuring of the process itself, culture of people involved, skills of actors, and process capabilities. Classically, the Sapir-Whorf hypothesis [914] or the “principle of linguistic relativity” postulates

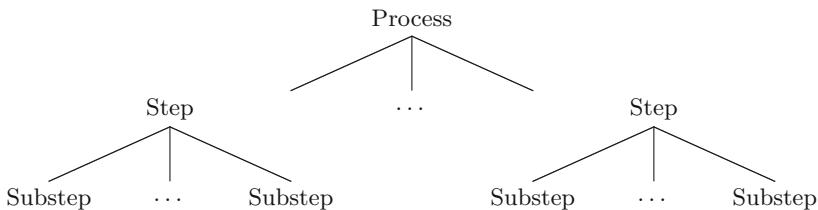


Fig. 12.2. Hierarchic structuring of activities

that actors skilled in a language may not have a (deep) understanding of some concepts of other languages.

Activities typically proceed in one or (usually) more steps. Activities are describing the ways how the work is performed and can be practices according to SPICE. A practice contributes to achieving a specific development process purpose or to the achievement of a specific development process attribute or enhances the capability of a development process. Activities consist of steps.

Steps are composed of other steps. We assume that steps can be hierarchically formed based on basic steps. Typical basic steps are the information seeking step, the collaboration step, the assessment step, the result integration step and the creation step. Steps can be composed to another step based on sequential, parallel, alternative, etc. composition operators. We may assume that steps can be also modelled through the specification language SiteLang on the basis of scenarios and stories.

Creation steps are the most complex steps. They typically consist of an orientation or review substep, of a development step performed in teams, and of a finalisation substep. Creation steps are composed of a number of substeps that can be classified into:

- Review of the state-of-affairs: The state of the development is reviewed, evaluated, and analysed. Obligations are derived. Open development tasks can be closed, rephrased or prioritised.
- Study of documents and resources: Available documents and resources are checked whether they are available, adequate and relevant for the current step, and form a basis for the successful completion of the step.
- Discussions and elicitation with other partners: Discussions may be informal, interview-based, or systematic. The result of such discussions is measured by some quality criteria such as trust or confidence. They are spread among the partners with some intention such as asking for revision, confirmation, or extension of the discussion.
- Recording and documentation of concepts: The result of the step is usually recorded in one work product or consistently recorded in a number of work products.
- Classification of concepts, requirements, results: Each result developed is briefly examined individually and in dependence of other results on which it depends and on which it has an impact.

- Review of the development process: Once the result to be achieved is going to be recorded the work product is examined for whether it has the necessary and sufficient quality, whether it must be revised, updated or rejected, whether there are conflicts, inconsistencies or incompleteness or whether more may be needed. If the evaluation results in requiring additional steps or substeps then the step or the substep is going to be extended by them.

This classification is based on the general problem solution framework discussed in [657]. Creation steps are usually iterative and thus cyclic as displayed in [Figure 12.3](#).

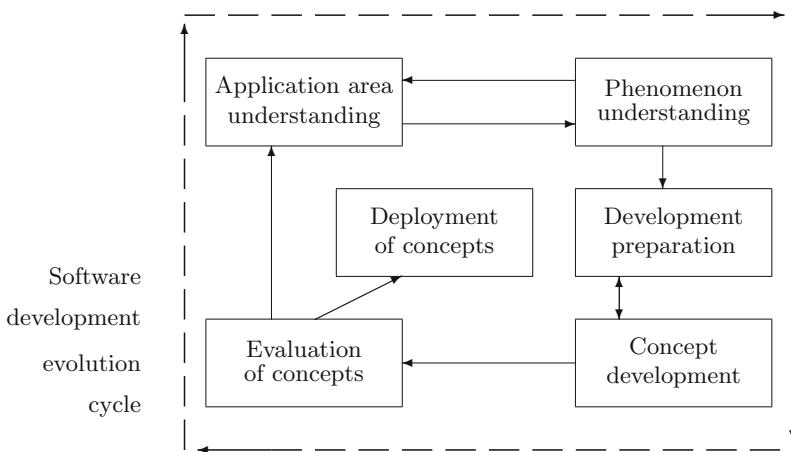


Fig. 12.3. The activity cycle during WIS development

The general frame to development is then based on the problem solution or search problem:

Problem characterisation with abstracting from non-essential parts;
Context injection for simplification of the problem and of the solution;
Tools and instruments for solution based on constructors, associations, collections, and classification;
Specification/prescription/description as the results of the development problem.

Analysis and concept formation and development are two specific activities that depend on each other. Development aims in forming concepts as well as discovering inconsistencies and conflicts and resolving incompleteness. Concept formation is based on the mapping of properties, requirements or phenomena into WIS concepts. Concepts may also be alternative concepts. These alternative concepts can be used at a later step instead of the given

one. Typical techniques for concept development and formation are exploration and experimentation, skeptical evaluation, conjecturing and refuting. Additional techniques are investigative ones that use resources.

In the evaluation phase we show whether the development result is correct, and if so what kind of tests, etc. ought to be devised. We do not require completeness since this is a relative issue.

Concept validation is the process of inspection of each concept with reference to concepts it is based on. It is based on the application area description and checks whether the right concepts are developed. Validation produces a report that constitutes the correctness and that produces the extensions and updates necessary for the base concepts.

Concept verification analyses concepts in order to ascertain whether what has been developed satisfies a number of obligatory properties. It checks whether the concepts developed so far are correct according to some correctness criteria and according to proof or verification techniques that are currently applicable. It follows the same general frame we already discussed for problem solution. Techniques may be informal, i.e., based on verbal arguments or tests, qualitative, i.e., based on abstraction and qualitative reasoning, or formal, i.e., based on model checking and proof techniques.

The Work Product Dimension

The work product is a result or deliverable of the execution of a process and includes services, systems (software and hardware) and processed materials. It has elements that satisfy one or more aspects of a process purpose and may be represented on various media (tangible and intangible).

The development of a model of the WIS is a process of concept formation. We describe the concepts of the application domain first roughly, then terminologise them, narrate them and possibly formalise them. This model is later analysed, verified, and validated. It can also be used to build up elements of a theory of the application domain. The aim is to create informative, descriptive, and analytic documents about the application domain.

The description of the application domain, the prescription of requirements, and the design of the systems is based on work products with the following desirable content:

- *Models* include the description of business users, of the terminology we use, and of the business processes. Resources we use for development are explicitly given. The glossary or terminology is going to be given in an explicit form. We target a consolidated description.
- *General information* consists of current situation, information demand and ideas, concepts and facilities, scope and span, assumptions and dependencies, and implicit and derivative intentions.
- *Analyses* contain the analysis and the concept formation, the application domain validation, the requirements and systems design verification, and

the theory. Analysis and the concept formation also describes inconsistencies found, conflicts observed, incompleteness assumptions, and resolutions. Validation is based on business user walkthroughs and resolution. Verification is based on model checking, theorems and proofs, and test cases and tests.

- *Technological context and environment information* consists of the client, channel, and server properties. These properties can be given through functional properties or quality criteria such as performance, dependability, and maintenance. They are extended by a characterisation of platforms used for development, demonstration, execution, and maintenance.
- *General development information* consists of the name, place, and data of specification, partners, the synopsis, contracts, and the teams involved (management, developers, client staff, and consultants).
- *General work product information* is based on a specification of the acquisition process (studies, interviews, questionnaires, and indexed description units) and the facets we tackle: intrinsic facets, management and organisation, rules and regulations, scripts, and human behaviour.

The level of detail may range from rough sketching, systematic narration till detailed documentations. We use this structure for all work products: stakeholder contract, requirements prescription, and systems documentation. It has the advantage of simplicity of mappings.

The work products exhibit the description and prescription of the application domain and its phenomena and concepts. Typical phenomena are hierarchical and compositional phenomena, denotable and computable concepts, contextual and state concepts, temporal and spatial phenomena. We may add properties of temporal behaviour such as continuity, discreteness, hybridicity, and chaos, evolution and configuration properties based on statics and dynamics, properties describing the human-computer interaction such as tangibility and intangibility, and finally dimensionality. A phenomenon is humanly tangible if it can be observed by humans. The work products are mainly texts and thus hampered by the one-dimensionality of the textual language. At the same time, applications are highly structured and multidimensional. We can use a plethora of diagrammatic notations. We need however to combine them with formal methods [110, 320].

Analyses of work products are based on validation and verification activities or on formal capability measures. We analyse work products based on four criteria:

- *Well-formed work products (formal capability)*: Each work product should follow a certain structuring and should contain the concepts we need for further development. Structuring of the stakeholder contract and of the systems documentation is well-settled in software engineering. We develop a number of templates for each of the descriptions, prescriptions and specifications and map the content of these to documents that depend on them. We may also add the semantic dimension to this syntactic treatment.

- *Satisfactory work products (verification):* Each work product is obliged to satisfy quality criteria such as correctness, completeness, consistency, stability, verifiability, modifiability, traceability, unambiguity, reliability, exactness, and accuracy. Correctness can only be checked relatively to known facts. Completeness depends on the needs, mission or intention of the WIS. Stability is relative to the deontic characterisation of necessary, useful, and optional properties. Verifiability depends on the existence of corresponding proof or model checking techniques. Modifiability is based on the solution of the frame problem for the work products, i.e., on a characterisation which part remains to be stable for activities and work steps. Traceability can be provided if the change history and causal relationship of changes is maintained. Unambiguity is satisfied if no inconsistency, vagueness, or double meaning remains in the work product. The last criterion requires to meet assumptions and dependencies.
- *Feasibility of development (validation):* We distinguish between technological feasibility and economic feasibility. Technological feasibility is based on an evaluation of progress and stability of technology currently on hand. Often it is interleaved with maintaining legacy applications and environments. Economic feasibility depends on the development, deployment and maintenance cost. For this reason, software engineering methods for cost estimation such as the function point method are extended to WIS.
- *Coverage of intentions (validation):* Intentions have to be weighted and are specified based on their four aspects: purpose, intent, time, and representation. Whether a WIS satisfies these intentions cannot be measured in a formal form. We may however evaluate in a fuzzy form to which extent purpose and intents are met.

We can develop a general structuring of work products. It consists of

- *general characteristics of work products* such as the work product identifier, the work product name, and the general description of the work product, and
- *meta-characteristics or docket characteristics of work products* such as author data, life-cycle data, meta-meta-data (version, date, author, (micro-) contribution), technical meta-data (format, size,...), usage meta-data (level of skills, understanding, ...), juridical meta-data (copyright, privacy, utilisation, ...), annotation meta-data (person, date, description), and classification meta-data (topic maps or ontology, purpose, description, keywords).

The Partner Dimension and Roles of Business Users

By a partner we understand a person or a group of persons, united somehow by their common interest in or their dependency on the application domain. By a partner perspective we understand the, or an, understanding of the application domain shared by a specifically identified business user group –

a view that may differ from one group to another in the same application domain. The believability of the application domain model depends on clear identification and liaising of the model issues. Partners may form a group or team within a collaboration. We distinguish perspectives of business users, of staff supporting the deployment of the WIS, and of stakeholders of the developer's house.

The role and the involvement of the partner is determined by the concern of the partner. Client executive and other upper-level management expect WIS to improve their competitiveness and position. Tactical and operational management base their expectations on management and organisational issues. Non-management staff expect support in their daily work and to their interfaces with the customer.

We distinguish two extreme groups of partners: Business users and market-driven partners. These two different groups have varying or even contradicting requirements and approaches. Their roles and their participation in the development process differ to a large extent. [Figure 12.4](#) shows a categorisation of partners.

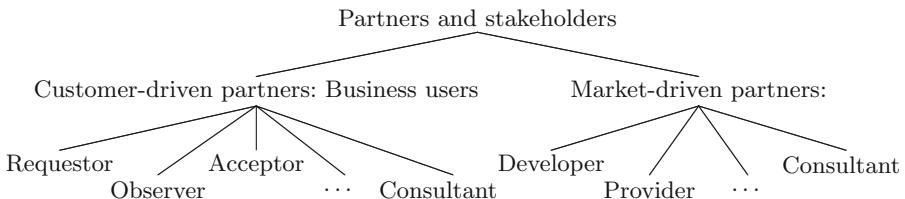


Fig. 12.4. Variety of roles of the partner

Business users have other primary interests than the projects which develop software or the products evolving from such projects. Typical business users can be characterised by their behaviour in the application domain such as transportation, manufacturing, mining, financial industries, public government, the service sectors, etc.

Aspects of the Development

We distinguish a number of facets or views on the application domain. Typical facets to be considered are business procedure and rule facets, intrinsic facets, support technology facets, management and organisation facets, script facets, and human behaviour.

Business processes, procedures and rules are procedurally describable ways in which business regularly conducts processes. They may include strategic, tactical, and operational processes. We combine life cases with business processes. The first are related to the portfolio, to the context, and to the intentions of business users. The latter describe how the business user acts, restrict

the behaviour of interest, and identify the resources necessary. They are based on content, functions, and events.

Intrinsic facets are common for all business users. Users may have different views or perspectives. They can be considered as the main entities, functions, and events to be supported. They are often mapped to components. For instance, modelling the schedule of trains and modelling the scheduling process itself use different views on the same application domain, the detailed view on the railway network, trains, and their operating. Usually, the basic concepts are not stated explicitly but must be isolated, identified, and captured. The basic units of the application domain are also called *utter barebones* [97].

The supporting technologies guide the implementation of all other facets. A WIS transforms the way in which users may act. At the same time we must be sure that supporting technologies are commensurate with our models of the application domain. The quality criteria expected in the application domain must be satisfied by the WIS.

The management and organisation in the application domain determine and constrain communication between users, the way of working, and requirements valid in the application domain. The management determines and sets standards or policies for operating. The organisation describes who does what, who uses which resources, and who reports to whom based on the administrative structure and on the roles the users play.

Application domain rules, regulations, and scripts guide the work and the interaction among users. They restrict business users in their behaviour and their involvement and guide the remedial actions for the case that a business user does not follow the intentions. Scripts are the most formal description and may have a legally binding power. They can be contested in a court.

We need to describe the way in which business users dispatch their actions and interactions. Typical such properties of human behaviour are careful, dutiful, diligent, accurate, forgetful, sloppy, or even criminal completion of actions. This behaviour style is captured by the profile of users and of actors. Arbitrariness is captured by the internal nondeterminism of the storyboard.

Facets are mapped to aspects of the development. These aspects concentrate either on models describing structural properties or evolution of the WIS or the collaboration among actors or the distribution or architecture of the system. [Figure 12.5](#) surveys aspects we consider.

Aspects describe different separate concerns.

Structuring of a database application is concerned with representing the database structure and the corresponding static integrity constraints.

Functionality of a database application is specified on the basis of processes and dynamic integrity constraints.

Distribution of information system components is specified through explicit specification of services and exchange frames.

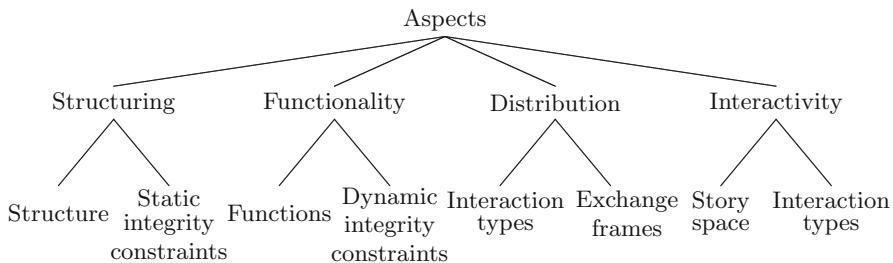


Fig. 12.5. Description, prescription and design aspects of the WIS

Interactivity is provided by the system on the basis of foreseen stories for a number of envisioned actors and is based on media objects which are used to deliver the content of the database to users or to receive new content.

This understanding has led to the *co-design approach* to modelling by specification *structuring*, *functionality*, *distribution*, and *interactivity*. These four aspects of modelling have both syntactic and semantic elements.

WISs are typically information-intensive systems. They may also support complex life cases and may become function-intensive. They are usually not event-intensive or process-intensive systems.

The Resource Dimension

Resources may be of different kinds as shown in [Figure 12.6](#). We use resources for enactment of steps. There are mandatory resources such as specification languages and methods. Most of the activities are also based on theoretical fundamentals.

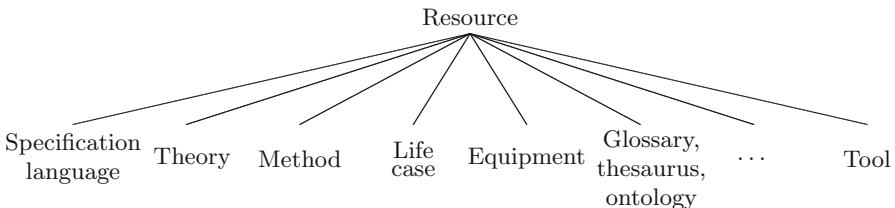


Fig. 12.6. Variety of resources to be used for WIS

The specification language must be powerful enough for covering all aspects, facets, etc. of the application. We may choose among a large variety of suitable languages. The language must have a formal foundation. It may also have a graphical or informal component. Since we are developing WIS we need to map all what is described informally or graphically to a formal notations.

Resources partially contain our knowledge on the theories used within the application domain. We do not need the entire and completely formalised theory of the application domain. Gathering and abstracting the theory might be a very difficult task. We can however start with the topics or annotations that are common in the application domain. Next we gain understanding, discover principles, discuss assertions, and reach some common sense with the stakeholders. Techniques for detection of the theory for the application domain are similar to those we use for modelling, i.e., gaining understanding, obtaining inspiration, presenting and becoming informed, asserting and predicting behaviour, and to capture restrictions of any kind.

Methods are given by principles, analysis, construction, technique, tool, and artifact. Principles provide an accepted or professed rule of action. Analysis may be performed through validation, verification and model checking. Construction is the process of creating a model. The techniques supporting this creation are based on transformations. During construction we may use tools. We use a formal framework for methods:

Method description: based on state changes in specification documents

Context for the method: artifacts of interest, input documents, dependencies, necessary information of higher quality

Applicability of the method: preconditions, application conditions, guards for application

Interaction with other methods: methods that depend on the given method, methods that must be completed before starting the current method, impact to other methods

Strategies of application: goals for application, intentions

Principles: rules for order of method application, requirements for accepting results of methods application

Analysis: tools for analysing the correct method application

Typical principles we consider are: typing of constructs, framed specification of concepts, known underlying theories (algebra, logics, etc.).

Life cases are used for specification of WIS pragmatics. They capture observations of user behaviour in reality. They are used in a pragmatic way to specify a story space, which is an important component of a storyboard.

Glossaries, thesauruses, dictionaries, and topic maps are used for the general annotation of concepts to be developed and refined. Topics are combined into topic maps. They generalise conceptual structures [798, 799], the Pawlak information model [643] and concept lattices [264].

Resource of the acquisition process are studies, legacy material, competitor products, interviews, questionnaires, indexes and other material that is available in the application domain.

Specification can be supported by equipment and tools. Equipment specifications are the basis for embedding the prescription into the specification. They restrict the feasibility of the requirements prescriptions and application domain descriptions.

12.1.3 Work Products of WIS Engineering

The WISE documentation is split into a number of specifications which are interrelated:

- *Work products used for application domain description* are HERM [823] concept maps, HERM functionality feature descriptions, the DistrLang distribution specification resulting in contract sketches which include quality criteria, and the SiteLang interactivity specification of the application story with main application steps.

The work products are the basis for the *Stakeholder contract* (“Lastenheft”). The stakeholder contract is extended by a rough project calculation and a rough project plan. Additional work products contain analysis of products of competitors, trend studies, marketing analysis, alternative solutions, summaries on previous developments, the glossary, quality properties, and studies on technological and economical feasibility. These work products are combined into a *feasibility study* product.

- *Work products used for requirements prescription* are the HERM sketch consisting of all main database types, the description of business processes with business steps, the DistrLang prescription consisting of the distribution contract opportunities, and the SiteLang story space consisting of main event and main story scenes.

The work products are combined within the *IS development and system documentation* (“Pflichtenheft”). The documentation is often extended by a general WIS architecture, decisions on the development environment, and by a prescription of the WIS environment.

- *Work products used for presentation system specification* describe the information and behavior conceived by the business user. Specification of structuring, functionality, distribution, and interactivity is based on the HERM skeleton with all application types, e.g., views on the data, on HERM activity diagrams describing the actions of business users, the specification of distribution made visible to the business user based on media types and contracts, and the specification of the SiteLang story space with the detailed description of the plot (theme), actors, media types, and of the presentation.

The work products are combined within the *Playout system specification*. Additionally, a user handbook can be sketched.

- *Work products used for systems specification* are refinements of the products developed so far. They consist of the HERM schema with fully developed types, the HERM workflow describing all processes, and the DistrLang schema for the service space and exchange frames.

The resulting work products and schemata are combined within the *Conceptual schemata*. The *architecture* of the envisioned solution and the separation of the schemata into components on the basis of a *Component contract schema* may be added in the case of implementation by collabor-

rating teams. The conceptual schemata are often extended by global test scenario that are going to be used within a test framework.

- *Work products of the implementation layer* are the (object-)relational schema using the data definition language of the target platform, the programming language modules, programs, triggers, stored procedures, the collaboration programs of the distributed system including the distribution architecture, protocols, and calls, and the dialog system programs of the presentation space including all working sheets supporting the business users.

This step results in an integrated *program library*, documentations, and deployment, maintenance, and evolution plans supporting all aspects of the envisioned system.

Work products are usually combined to consistent new products, refined to new products, updated, or abandoned. The operations used for the revision and creation of work products are similar to database operations. We thus do not present them in detail.

Work products may be given in a formal way or a visual form or in a narrative form. Similar to [Figure 11.17](#) we can use also diagrams or visualisation for the specification of work products. The worksheet in [Figure 12.7](#) describes the general realisation details for dialogue scenes.

dialogue scene	
header	
name	layout
title	
container	
content	
text	
multimedia object	graphics picture video audio
functionality	operations algorithmic object
tailoring style	
ordering by hierarchies	
adhesion	
adaptation	
interaction style	
control style	
actors applicable	

Fig. 12.7. Worksheet for general description of dialogue scenes

We might also use worksheets for an informative description of dialogue scenes. A typical small worksheet is given in [Figure 12.8](#). It provides an informative description for *explanation, informed selection, and appropriation* and shows which elements are mandatory, which elements are good practice or optional, and which additional elements might be considered to be useful.

scene			
header			
content problem area solution	name intention	developer motivation also known as	copyright source see too
variants	application area		
application			
applicability	consequences of application	sample applications	known applications
usability profile	experience reports	DBMS	
description			
structuring: static constraints	functionality: operations, dynamic constraints, enforcement procedures	interactivity: story space, actors, media objects, representation	context: tasks, intention, history, environment, particular
implementation			
implementation	code sample	associated scenario	
associated scenes	collaboration	integration strategy	
mandatory	good practice	optional	useful

Fig. 12.8. Worksheet for informative description of dialogue scenes

Controllers for WIS Development

We can identify two extremes of WIS development:

Turnkey development is typically started from scratch in a response to a specific development call.

Commercial off-the-shelf development is based on software and infrastructure whose functionality is decided upon by the makers of the software and the infrastructure rather than by the customers.

A number of software engineering models have been proposed in the past: waterfall model, iterative models, rapid prototyping models, V-model, Vienna development methods, extreme development, agile development, etc. The co-design approach is not bound to any of them. It can be integrated with all these methods.

At the same time, developers need certain flexibility during WIS engineering. Some information may not be available. We need to consider feedback loops for redoing work that has been considered to be complete. All dependencies and assumptions must be explicit in this case.

The co-design approach uses a development process model that consists of all primary dimensions shown in [Figure 12.1](#). The main work product used for the controller is a blackboard that uses a set of data structures.

Process documentation: Each development process is documented by documents that clarify the outcome according to [Figure 12.1](#).

Purpose and goals: The process is described by statements and the purpose and goals.

OUTCOMES, WORK PRODUCTS: Work products are going to be revised or created and completed during the current process.

Resources: Processes use a number of resources such as work products, theories, models, application domain description, auxiliary resources, etc.

Steps are elements of the activities. They correspond to base practices in SPICE and CMM.

Detailed description: Steps are performed by partners, use resources, are oriented to refine or create a number of work products.

Conditions: Steps can only be started if preconditions are fulfilled. They can be only considered to be completed if all post-conditions are valid. Post-conditions can be categorised to accept-on conditions that describe obligations before completion, general post-conditions that clarify which state change in the work products must be observed, rely-conditions which state changes in the work products cannot appear, and guarantee-conditions which specify quality improvements during step execution. The conjunction of pre- and post-conditions is often called invariant.

Aspects: The co-design framework is based on separation of concern. Each of the steps will enlighten only some of the aspects.

Work products: Work products may be used, revised, refined, or abandoned during steps.

Partner: Steps are performed in a collaboration among partners. Partners play certain roles, have a certain involvement in the WISE and a portfolio.

Obligations: The overall WISE aims in development of a complete WIS specification and WIS implementation. The requirements for the WISE can be recorded in an obligations log. Requirements are marked as planned, started, under review, and completed. Steps may add obligations due to the outcome of the step.

The controller model must also support analysis of work products in dependence of quality criteria. We may develop a matrix representing the relationship between work product, quality criterion and analysis technique to be

applied. Another presentation technique is presented in [Figure 12.9](#). Security is a criterion that should be verified for the HERM sketch. Reliability is verified both for the SiteLang story space and the HERM sketch. Productivity is analysed based on model checking techniques for the SiteLang story space.

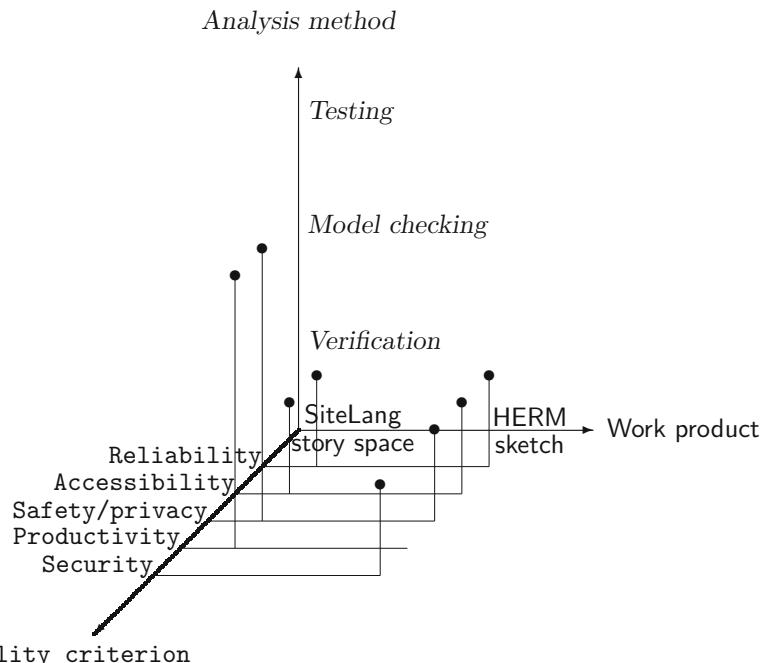


Fig. 12.9. Controlling quality for requirements prescription work products

12.1.4 The Relationship of WIS Engineering to SPICE

SPICE uses a two-dimensional, continuous assessment model. This means that processes and capabilities are in separate dimensions, which provides that each process can be separately assessed and its capability determined. The middle and right regions in [Figure 12.10](#) show the principal elements of SPICE and how they are related to each other.

The process dimension defines 48 *processes*, which are classified by nine *process groups*: Acquisition, Supply, Operation, Engineering, Supporting, Management, Process Improvement, Resource and Infrastructure, and Reuse. The indicators of process performance are process specific *base practices and work products*. The left column of Table 1 lists three of the twelve Engineering processes with their base practices.

The capability dimension consists of six *capability levels*: Incomplete, Performed, Managed, Established, Predictable and Optimizing. The capability

is measured based on *process attributes*. The indicators of process capability are *generic practices*, *generic resources* and *generic work products*. For example, the process attributes on Level 2 are 1) Performance management and 2) Work product management, and respectively on Level 3 1) Process definition and 2) Process deployment.

The co-design framework defines aspects, layers and steps, and documents related to the layers. These concepts can be interpreted with generic process concepts, i.e., processes, practices and work products (Figure 12.10). The evaluation process here is a quite similar to process assessment [543, 544] but is based only on the assessors' interpretation of co-design documentation. This limits the reliability of the study and therefore attribute level ratings are not presented. On the other hand, exact capability determination is not a goal of the evaluation; instead we concentrate on identifying relevant improvement opportunities.

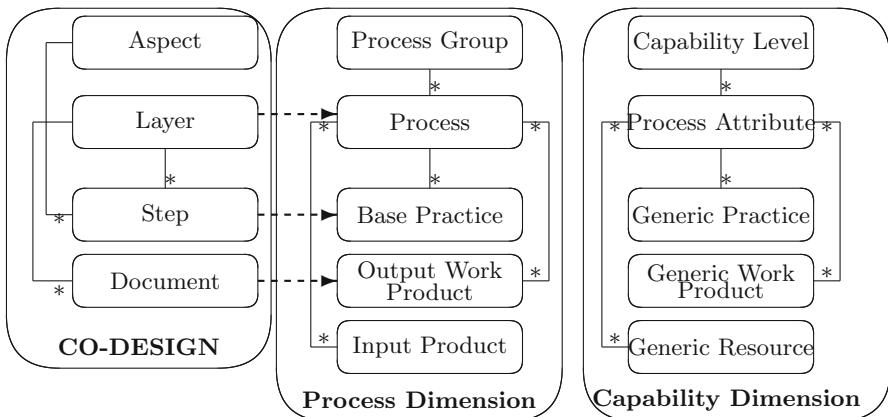


Fig. 12.10. The principal element types of co-design, SPICE process dimensions and SPICE capability dimensions

12.1.5 Orchestration of WIS Development for Managed Engineering

Orchestration uses the metaphor to music. It denotes the arrangement of a musical composition for performance by an orchestra. It may also denote harmonious organization, i.e., through orchestration of cultural diversities. Figure 12.1 displays dimensions of information systems engineering. Partners must integrate all activities, for all aspects, for all resources, and all work products. SPICE [347] requires for managed development processes that the

implemented process achieves its process purpose (SPICE level 1). SPICE level 2 is achieved if the process is well-specified and is implemented in a managed fashion (planned, monitored and adjusted) and its work products are appropriately established, controlled and maintained.

Therefore, managed engineering is based on performance management and on work product management. Performance management requires that objectives for the performance of the process are identified, performance of the process is planned and monitored, performance of the process is adjusted to meet plans, responsibilities and authorities for performing the process are defined, assigned and communicated, resources and information necessary for performing the process are identified, made available, allocated and used, and interfaces between the involved parties are managed to ensure both effective communication and also clear assignment of responsibility. Work product management requires a well-defined and well-implemented work product management. Requirements for the work products of the process must be defined as well as requirements for documentation and control of the work. Work products must be appropriately identified, documented, and controlled. Work products are going to be reviewed in accordance with planned arrangements and adjusted as necessary to meet requirements.

12.1.6 Evolving the Co-Design Framework by SPICE

Languages used in the co-design framework are intensively discussed in [823]. These languages have been widely used in a number of industrial projects and are the basis of design workbenches such as ID² and (DB)². A variety of methodologies to development of information systems can be derived [833]. The co-design methodology has also been used for reengineering of groups of database applications, e.g., the universal data model resource [826] for product applications. Most of them are based on top-down or refinement approaches that separate aspects of concern into abstraction layers and that use extension, detailisation, restructuring as refinement operations.

The classical co-design approach to information systems development can be layered. Steps in these layerings are:

Motivation layer

1. Developing visions, aims and goals
2. Analysis of challenges and competitors

Strategic layer

3. Separation into system components
4. Sketching the story space
5. Sketching the view suite
6. Specifying business processes

Business user layer

7. Development of scenarios of the story space

8. Elicitation of main data types and their associations
9. Development of kernel integrity constraints, e.g., identification constraints
10. Specification of user actions, usability requirements, and sketching media types
11. Elicitation of ubiquity and security requirements

Conceptual layer

12. Specification of the story space
13. Development of data types, integrity constraint, their enforcement
14. Specification of the view suite, services and exchange frames
15. Development of workflows
16. Control of results by sample data, sample processes, and sample scenarios
17. Specification of the media type suite
18. Modular refinement of types, views, operations, services, and scenes
19. Normalization of structures
20. Integration of components along architecture

Implementation layer

21. Transformation of conceptual schemata into logical schemata, programs, and interfaces
22. Development of logical services and exchange frames
23. Developing solutions for performance improvement, tuning
24. Transformation of logical schemata into physical schemata
25. Checking durability, robustness, scalability, and extensibility

The separation of the co-design framework within the SPICE levels has resulted in

- the development of light-weight co-design frameworks which conforms with SPICE level (1)
- the development of approaches for reaching SPICE level (4) including measures, measuring attributes, and process control attributes for measuring the development of specification outcome,
- the comparison of the SPICE and co-design frameworks targeting in development of proposals for enhancement of the SPICE framework.

The evaluation of the Co-Design framework showed that the framework [833] highly supports information system development in achieving SPICE level 1 of the selected processes. It is our aim to evolve the co-design framework to SPICE level (2). In order to reach this aim we extended the specification frame of each step to the following specification frame:

Step 1.	
Name of the step with abstraction layer or as orthogonal step to abstraction layer	Task 1. Task 2. ...
Used documents	Documents of previous steps (IS development documents) Customer documents and information
Documents under change	IS development documents Contracts
Aims, purpose and subject	General aims of the step Agreed goals for this step Purpose Matter, artifact
Actors involved	Actor A, e.g., customer representatives Actor B, e.g., developer
Theoretical foundations	Database theory Organization theory Computer science Cognition, psychology, pedagogics
Methods and heuristics	Syntax and pragmatics Used specification languages Simplification approaches
Developed documents Results	IS development documents Results and deliverables
Enabling condition for step	Information gathering conditions fulfilled by the customer side Information dependence conditions Conditions on the participation
Termination condition for step	Completeness and correctness criteria Sign-offs, contracts Quality criteria Obligation for the step fulfilled

The specification frame is currently revised for extension to SPICE level (3) processes. The next step is the refinement to predictable development processes that guarantee quality of the documentation. Suitability, understandability, learnability, attractiveness are quality criteria at the motivation and strategic layers. Operability and analysability are criteria at the business process and the strategic layers. Security, appropriateness, and accuracy are additionally required at the business user layer. Interoperability, changeability, stability, testability, and installability are the main quality criteria at the conceptual layer. Robustness, flexibility, maturity, fault tolerance, recoverability,

availability, time behavior, resource utilization, adaptability, and co-existence replaceability are targets at the implementation layers.

Finally, the co-design framework may be developed in its full unfolding or as lightweight co-design for WIS. [Figure 12.11](#) compares the work products in both approaches.

Step	Aspects				Lightweight co-design	
	Structuring	Functionality	Distribution	Interactivity	Engineering of small database systems	Engineering of web information systems
Visions	✓	✓	✓	✓	-	✓
Stakeholder contract	✓	✓	✓	✓	(✓)	-
Application architecture	✓	✓	✓	-	-	-
Development of stories	-	-	-	✓	-	✓
View documentation	-	-	✓	-	✓	-
Business processes	-	✓	-	-	✓	-
Detailed scenarios	-	-	-	✓	-	✓
Kernel type	✓	-	✓	-	✓	-
Identification frames	✓	✓	-	-	✓	-
Actions	-	-	-	✓	-	✓
Interaction diagrams	-	(✓)	-	✓	✓	✓
Finalizing type specification	✓	✓	-	-	✓	✓
Handling identification	✓	✓	-	-	-	-
Sample data	✓	✓	-	-	-	-
Storyboard development	-	-	-	✓	-	✓
Normalization	✓	-	-	-	✓	-
Domain types	✓	-	-	-	-	-
Functionality	-	✓	-	-	(✓)	✓
View integration	✓	-	✓	-	(✓)	-
Incremental transformation	✓	✓	✓	✓	✓	✓
Optimization	✓	✓	✓	✓	-	-
Performance of huge databases	✓	-	-	-	-	-
Potential changes	✓	-	-	-	-	-
DBMS evolution	✓	✓	✓	✓	-	-
Distributed database systems	✓	-	✓	-	-	-
Database farms	✓	✓	✓	-	-	-
Incremental database systems	✓	-	✓	-	-	-
Database warehouses	✓	-	✓	-	-	✓
Security concepts	-	-	✓	-	-	✓

Fig. 12.11. Tasks and work products in the full and in the lightweight co-design framework

12.2 Architectures of Web Information Systems

There is no commonly agreed definition for the notion of a software architecture.¹ Some of the notions we found in the literature are too broad, some others are too narrow.² We follow the approach in [359] and use the following definition of the notion of an architecture:

A system architecture represents the conceptual model³ of a system together with models derived from it that represent (1) different viewpoints defined as views on top of the conceptual model, (2) facets or concerns of the system in dependence on the scope and abstraction level of various stakeholders, (3) restrictions for the deployment of the system and description of the quality warranties of the system, and (4) embeddings into other (software) systems.

12.2.1 An Architectural Framework

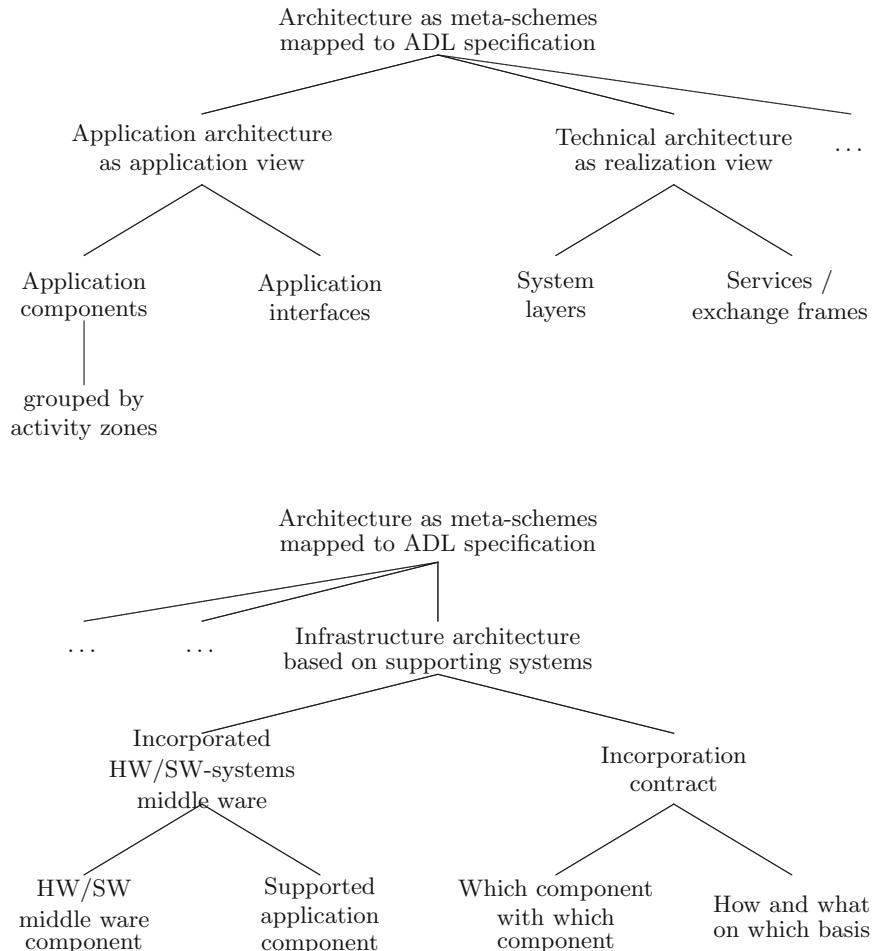
We can distinguish five standard views in an architectural framework [359]:

- (I) The *information* or *data view* represents the data that is required by the business to support its activities. This answers the what information is being processed question.
- (II) The *functional business* or *domain view* represents all the business processes and activities that must be supported. This answers the “what business activities are being carried out”.
- (III) The *integration* or *data-flow view* represents the flow of information through the business, where it comes from and where it needs to go. This answers the which business activities require it question.
- (IV) The *deployment* or *technology view* represents the physical configuration and technology components used to deploy the architecture in the operating environment. This answers the where is the information located question.
- (V) The *infrastructure* or *embedment view* represents the system as a black- or grey-box and concentrates on the embedding of the system into other systems that are either supporting the system or are using systems services.

¹ Compare the large list of more than hundreds of definitions collected from contributors to <http://www.sei.cmu.edu/architecture/start/community.cfm>

² Compare <http://www.sei.cmu.edu/architecture/start/moderndefs.cfm>

³ The conceptual model includes structural, behavioural and collaboration elements. Systems might be modularised or can also be monolithic. The conceptual model allows us to derive a specification of the system capacity. We may distinguish between standard views and views that support different purposes such as system construction, system componentisation, documentation, communication, analysis, evolution or migration, mastering of system complexity, system embedding, system examination or assessment, etc.

**Fig. 12.12.** Architecture of WIS

Web information systems are typically layered or distributed systems. Layering and distribution results in rather specific data structures and functions that are injected in order to cope with the specific services provided by layers or components. The *CottbusNet* projects used a multi-layer and distributed environment. For instance, the events calendar in city information systems may use a dozen or more different database systems and a view tower. A view tower of such systems must provide advanced search facilities [204]. It uses views that compile a variety of ETL results into a common view for events, an extraction view for presentation of events at a classical website or at other media such as video text canvas or smart phone display, a derived search functionality for these data, and a collection view for a shopping cart of an

event shopper. A similar observation can be made for OLTP-OLAP systems [496, 501]. OLAP systems are typically built on top of OLTP systems by applying first grouping and aggregation functions and second by integrating data obtained into a data mart presentation.

The WIS architecture sketches the overall WIS by separating concerns, by clustering parts, by determining an overall component structuring and the corresponding interfaces and by separated treatment of each of the components. Architecture development aims at decomposition of requirements into subsystems. This decomposition results in component development and in definition of exchange frames between these components.

The application architecture is the abstraction on the architecture that shows the playout of the WIS depending on the story space. The quality criteria, their priority, and their analysis are combined in an *application architecture*. The WIS is shown from the viewpoint of the application. Constituents of the application architecture are surveys on the components envisioned, their behaviour, services and interfaces. Components are abstractly specified in a black-box form. The inner structure is not provided.

The *technical architecture* provides a description of components of the WIS that are necessary for operating the WIS. Layering of services or presentation via multi-tiered architectures supports also analysis, e.g., treatment of errors.

The *architecture of the technical infrastructure* is oriented towards a presentation of those hardware and software components and their communication models that are integrated into the WIS. Classically all peripheral devices, system software, specific protocols are prescribed.

Architectural styles provide an abstract description of general characteristics of a solution. [Table 12.1](#) list some of the styles [359].

Each of these styles has strengths, weaknesses, opportunities, and threats. Strengths and opportunities of certain architectural styles are widely discussed. Weaknesses and threats are discovered after implementing and deploying the decision. For instance, the strengths of SOA (service oriented architecture) are domain alignment, abstraction, reusable components, and discoverability. Weaknesses of SOA are acceptance for SOA within the organization, harder aspects of architecture and service modeling, implementation difficulties for a team, methodologies and approaches for implementing SOA, and missing evaluations of various commercial products that purport to help with SOA rollouts. Threats of SOA are the development of a proper architectural plan, the process plan, resource scope, the application of an iterative methodology, the existence of a governance strategy, and the agreement on clear acceptance criteria.

Therefore, a selection of an architecture has a deep impact on the web information system itself and drives the analysis, design and development of such systems. [Figure 11.7](#) considers a separation of systems into a presentation system and a support system, i.e., the classical client-server decision. The picture is more complex if we decide to use 3-tier, SOA or other architectures. The structuring and the functionality that are provided by each of the sub-

Table 12.1. Description of architectural styles

Style	Description
Client-Server	Segregates the system into two applications, where the client makes a service request to the server.
Component-Based Architecture	Decomposes application design into reusable functional or logical components that are location-transparent and expose well-defined communication interfaces.
Layered Arch.	Partitions the concerns of the application into stacked groups (layers).
Message-Bus	A software system that can receive and send messages that are based on a set of known formats, so that systems can communicate with each other without needing to know the actual recipient.
N-tier / 3-tier	Segregates functionality into separate segments in much the same way as the layered style, but with each segment being a tier located on a physically separate computer.
Object-Oriented	An architectural style based on division of tasks for an application or system into individual reusable and self-sufficient objects, each containing the data and the behavior relevant to the object.
Separated Presentation	Separates the logic for managing user interaction from the user interface (UI) view and from the data with which the user works.
SOA	Refers to applications that expose and consume functionality as a service using contracts and messages.

systems must be properly designed. Therefore, the architectural style is going to drive the development process.

12.2.2 Architecture-Driven Development

WIS specification is often based on an incremental development of WIS components, their quality control and their immediate deployment when the component is approved. The development method is different from those we have used in the first layers. Application domain description aims in capturing the entire application based on exploration techniques. Requirements prescription is refining the application domain description. Specification is based on incremental development, verification, model checking, and testing. This incremental process leads to different versions of the WIS: *demo WIS*, *skeleton WIS*, *prototype WIS*, and finally *approved WIS*.

Software becomes surveyable, extensible and maintainable if a clear separation of concerns and application parts is applied. In this case, a skeleton of the application structure is developed. This skeleton separates parts or services. Parts are connected through interfaces. Based on this architecture blueprint, an application can be developed part by part.

We *combine* modularity, star structuring, co-design, and architecture development to a novel framework based on components. Such combination

seems to be not feasible. We discover, however, that we may integrate all these approaches by using a component-based approach [831, 834]. This skeleton can be refined during evolution of the schema. Then, each component is developed step by step. Structuring in component-based co-design is based on two constructs:

Components: Components are the main building blocks. They are used for structuring of the main data. The association among components is based on ‘connector’ types (called hinge or bridge types) that enable associating the components in a variable fashion.

Skeleton-based construction: Components are assembled together by application of connector types. These connector types are usually relationship types.

The architecture description language (ADL) is a template for architecture depiction and consists of the following parts:

Architecture Modelling Features

- Components
- Interface
- Types
- Semantics
- Constraints
- Evolution
- Non-functional properties

Connectors

- Interface
- Types
- Semantics
- Constraints
- Evolution
- Non-functional properties

Architectural Configurations

- Understandability
- Compositionality
- Refinement and traceability
- Heterogeneity
- Scalability
- Evolution
- Dynamism
- Constraints
- Non-functional properties

Tool Support

- Active Specification
- Multiple Views
- Analysis

Refinement
Implementation Generation
Dynamism

The separation of the WIS into components supports stepwise and incremental development of the WIS. The separation is based one ordering concepts and interrelating them properly. Incremental development eases refinement, stepwise reification, and stepwise transformation. This approach also supports controlled refinement based on a refinement strategy. The choice of refinements is determined so as to satisfy those application domain requirements which were not all taken care of or which were only partially taken care of in previous steps.

This stepwise unfolding of architecture during WIS development takes care of one concern. We can also interweave layers of development in the co-design approach as long as no additional interdependencies are created. Separation allows to sort out those parts of the system that remain to be invariant. The development is controllable by scoping and localisation abstraction. This separation is one basis for tracking later development steps to more early ones. It is also the basis for analysis of the work products. Correctness proofs are easier to apply.

Main architectural structure are components and connectors. Components are computational elements. Ports are interface points for components. Connectors support interactions between components. We assign roles to interface points for connectors. Abstraction mappings relate the inside and outside of architectural views. Bindings are a special case of such mappings.

The general WIS architecture results in a component model. The component consists of a suite of modules that is relatively independent of other modules. Each component has a set of interfaces. The interface can be used for a well-specified data exchange between components.

A WIS component is a media type, i.e., has a structure and functions, obeys static and dynamic integrity constraints, and has a set of interfaces that are defined as views on the structure of the component. Views can be used for retrieval requests, for manipulation of data, or for control. Whether the component can be updated or not depends on the structure of the component and the static integrity constraints. Views can be refined or extended.

Components can be connected with each other. The connection is based on the exchange frame and especially the coordination contract. The latter restricts concurrent utilisation of interfaces for instance through ACID coordination and rules for repeated reads (ranging from unrestricted, to restricted or to non-repeatable). Components may be loosely or tightly coupled. Coupling is a dynamic function. The coupling is controlled by the collaboration contract. Collaboration contracts generalise connector protocols which usually consist of a set of traces of input/output behaviours.

The connector is modelled by a channel which is a specific container. The container has a number of properties restricting exchange of data with the container.

12.2.3 Pattern-Based Development

A typical engineering approach to development of large conceptual models is based on general solutions, on an architecture of the solution and on combination operations for parts of the solution. We may use a two-layer approach for this kind of modelling. First, generic solutions are developed. We call these solutions *conceptual schema pattern set* [17]. The architecture provides a general *development contract* for subparts of a schema under development.

The theory of conceptual modelling may also be used for a selection and development of an assembly of modelling *styles and perspectives*. Typical well-known styles [823] are inside-out refinement, top-down refinement, bottom-up refinement, modular refinement, and mixed skeleton-driven refinement. A typical perspective is the three-layer architecture that uses a conceptual model together with a number of external models and an implementation model. Another perspective might be the separation into an OTP-OLAP-DW system. The adaptation of a conceptual schema pattern set to development contracts and of styles and perspectives leads to a *conceptual schema grid*.

12.2.4 Architecture Blueprint

An architecture blueprint [359] consists of models, documents, artifacts, deliverables, etc. which are classified by the following states:

The architecture framework consists of the *information or data view*, *functional business or domain view*, *integration or data-flow view*, *deployment or technology view*, and *infrastructure or embedment view*.

The WIS development architectures guide: The *current architecture* is the set of all solution architecture models that have been developed by the delivery projects to date. Ownership of the solution architecture models are transferred to the current Enterprise Architecture when the delivery project is closed. The *development state architecture* represents the total set of architecture models that are currently under development within the current development projects. The *target vision state architecture* provides a blueprint for the future state of the architecture needed in order to satisfy the application domain descriptions and target operating model.

12.2.5 The CottbusNet Design and Development Decisions

Let us consider the event calendar in an infotainment setting of a city information system. This calendar must provide a variety of very different information from various heterogeneous resources:

- Event-related information: Which event is performed by whom? Where from are the actors? How is the event going on?
- Location-based information: Which location can be reached by which traffic under which conditions with whose support?
- Audience information: Which audience is sought under which conditions, regulations and with which support?
- Marketing information: Which provider or supplier markets the event under which time restrictions with which business rules?
- Time-related information: Which specific time data should be provided together with events?
- Intention information: Are there intentions of the event that should be provided?

The event calendar is based on different databases: event databases for big events, marketing events, sport events, cultural events, minor art events, etc.; location databases for support of visitors of the event providing also traffic, parking, etc. information; auxiliary databases for business rules, time, regulations, official restrictions, art or sport activists, reports on former events, etc.

It is not surprising that this information is provided by heterogeneous databases, in a variety of formats, in a large bandwidth of data quality, in a variety of update policies. Additionally, it is required to deliver the data to the user in the right size and structuring, at the right moment and under consideration of the user's information demand. Consider, for instance, minor art events such as a cabaret event held in a restaurant. The information on this event is typically incomplete, not very actual, partially inexact and partially authorised. The infotainment site policy requires however also to cope with such kinds of events.

We may consider now a number of architectures, e.g., the following ones:

- *Server-servlet-applet-client layered systems* typically use a *ground database system* with the production data, a number of *serving databases systems* with the summarised and aggregated data based on media type technology [720], and *playouting systems* based on container technology [501] depending on adaption to the storyboard [722].
- *OLTP-OLAP-Warehouse systems* [495, 496] use a ground database system for OLTP computing, a derived (summarised, aggregated) OLAP system for comprehensive data delivery to the user, and a number of data warehouses for data playout to the various kinds of users.

Depending on these architectures we must enhance and extend the conceptual schema for the different databases, the workflow schemata for data input, storage, and data playout to the user.

12.3 WIS Development Dimensions

12.3.1 Primary WIS Development Dimensions

The seven *primary dimensions of the development process* [174] (see Chapter 11) use the classical rhetorical frame introduced by Hermagoras of Temnos⁴ (Quis, quid, quando, ubi, cur, quem ad modum, quibus adminiculis (W7: Who, what, when, where, why, in what way, by what means)). These primary dimensions are an integral part of each development step. WIS will also be used over decades whereas software and hardware changes more often. For this reason, the complete information systems development process must be well-documented. The entire development knowledge that is necessary for building, using, and maintaining information systems must be kept over the lifetime of the system. The documentation includes meta-data on the enterprise system, the technical process supporting the system, the business processes supported by the system, services and software tools used within the system, organizational policies and people.

12.3.2 Secondary WIS Development Dimensions

The secondary dimensions include a number of auxiliary stages which should be integrated into the development process: application domain knowledge acquisition, analysis and concept formation, verification, validation, and application domain theory formation.

Typical secondary dimensions in the W*H description frame [175] are:

Competency: All tasks collected in the portfolio of users are explicitly supported by the WIS.

Time: Data such as prescription, description and specification data are aging too. The aging is caused by changes in the deployment of the WIS, by changes in views, by changes in the content, and by changes within the infrastructure used. We often used auxiliary temporal structures for representation of this dimension.

Environment: The environment determines a large number of design decisions and should be explicitly taken into account. The co-design framework distinguishes task context, actor context, goals context, WIS context, legacy context, temporal context, and spatial context. Additionally we might also consider the organisational context and the provider context.

⁴ His work is almost lost. It is however reflected in the work of his followers, e.g., Cicero. Before, Aristotle had already used topics, foci, viewpoints, styles, and scopes for rhetoric functions. Far later in computer engineering J. A. Zachman is considered to be the inventor of this framework and of foci of models despite the well-known frame used since two thousand years.

Quality: A number of functional and non-functional quality properties such as accuracy, security, consistency, interpretability, robustness, scalability, and durability should be taken into account for each of the development steps. We may use a framework supporting quality management or may use evaluations that determine which of the quality properties is (partially) satisfied in the development.

Runtime characteristics: The system environment and the user environment are also influenced by exception situations and delays. Additionally, adaptation to the user, especially the way how a user performs a task, and the currently used environment change the run of a system.

Collaboration: WISs are often supporting groups of users. Therefore collaboration aspects such as communication, cooperation, and coordination are modelled explicitly and integrated with the other specifications.

Motivation and intentions: The motivation of user for WIS use is explicitly specified through four aspects of intentions: purpose (aims or objectives), intents (targets or objects), time (design, end, or occasion), and representation aspects (atmosphere or metaphors).

Meta aspects [457] are not considered well in most WIS descriptions. Typical meta aspects are runtime, operational, provenance and quality aspects characterising for instance the work products themselves.

12.3.3 The Quality of WIS

Quality of WIS is characterized depending on the abstraction layers [358, 359]:

- *Quality parameters at business user layer* may include *ubiquity* (access unrestricted in time and space) and *security/privacy* (against failures, attacks, errors; trustworthy; privacy maintenance).
- *Quality parameters at conceptual layer* subsume *interpretability* (formal framework for interpretation) and *consistency* (of data and functions).
- *Quality parameters at implementation layer* include *durability* (access to the entire information unless it is explicitly overwritten), *robustness* (based on a failure model for resilience, conflicts, and persistency), *performance* (depending on the cost model, response time and throughput), and *scalability* (to changes in services, number of clients and servers).

We use a number of measures that define quality of service (QoS) for WIS:

- *Deadline Miss Ratio of User Transactions:* In a WIS QoS specification, a developer can specify the target deadline miss ratio that can be tolerated for a specific real-time application.
- *Data Freshness:* We categorize data freshness into database freshness and perceived freshness. Database freshness is the ratio of fresh data to the entire temporal data in a database. Perceived freshness is the ratio of

fresh data accessed to the total data accessed by timely transactions - transactions which finish within their deadlines.

- *Overshoot* is the worst-case system performance in the transient system state. In this paper, it is considered the highest miss ratio over the miss ratio threshold in the transient state. In general, a high transient miss ratio may imply a loss of profit in e-commerce.
- *Settling time* is the time for the transient overshoot to decay and reach the steady state performance.
- *Freshness of Derived Data*: To maintain the freshness, a derived data object has to be recomputed as the related ground database changes. A recomputation of derived data can be relatively expensive compared to a base data update.
- *Differentiated Timeliness*: In WIS QoS requirements, relative response time between service classes can be specified. For example, relative response time can be specified as 1:2 between premium and basic classes.

We observe that these quality of services characteristics are difficult to specify in systems if architecture is not taken into consideration. Let us consider data freshness as an example for WIS. Data freshness results are related to *information logistics* that aims in providing the correct data at the best point of time, in the agreed format and quality for the right user at the right location and context. Methods for achieving the logistics goals are the analysis of the information demand, storyboarding of the WIS, an intelligent information system, the optimization of the flow of data and the technical and organizational flexibility. Therefore, data freshness can be considered to be a measure for appropriateness of the system. Depending on the requested data freshness we derive the right architecture of the system.

Based on our co-design modelling approach and as a result of separation of concern within the software engineering quadruple we can derive a number of techniques for architecture-driven and application-domain-rules modelling of high quality WIS:

- *Introduction of artificial bottlenecks*: Instead of replicating data at different sites or databases we may introduce a central data store that exhibits a number of versions to each of the clients that require different data.
- *Introduction of a tolerance model*: We may introduce an explicit tolerance model that decreases the burden of data actuality to those web pages for which complete actuality is essential.
- *A cost-benefit model of updates*: Updates may sometimes cause a large overhead of internal computing due to constraint maintenance and due to propagation of the update to all derived data. We thus may introduce delays of updates and specific update obligations for certain time points. Typical resulting techniques are *dynamic adaptation of updates* and the explicit treatment by an *update policy*.

- *Data replication in a distributed environment:* Data access can be limited in networking environments. The architecture may however introduce explicit data replication and specific update models for websites.

This list of techniques is not complete but demonstrates the potential of architecture-driven WIS development.

12.3.4 The Semiotics Background and Pragmatism

Conceptual modelling uses languages. Therefore, we are bound to the conceptions of the language, the expressivity of the language, and the methodology for language utilisation. The Sapir-Whorf observation [914] postulates that developers skilled in a certain language may not have a (deep) understanding of some concepts of other languages.

Semiotics distinguishes between syntactics (concerned with the syntax, i.e., the construction of the language), semantics (concerned with the interpretation of the words of the language), and pragmatics (concerned with the meaning of words to the user). Most languages used in computer science have a well-defined syntax, their semantics is provided by implementations which are often not made public and their pragmatics is left to experimentation by the user. This ‘banana’ or ‘potato’ principle leads to very different utilization and does not allow consistent use by groups of developers and over time. Each step is based on a specification language that has its syntax, semantics, and pragmatics.

Syntactics: Inductive specification of structures uses a set of base types, a collection of constructors and a theory of construction limiting the application of constructors by rules or by formulas in deontic logics. In most cases, the theory may be dismissed. **Structural recursion** is the main specification vehicle. Constructors may be defined on the basis of grammatical rules.

Semantics: Specification of admissible databases on the basis of static integrity constraints describes those database states which are considered to be legal. If structural recursion is used then a variant of hierarchical first-order predicate logics may be used for description of integrity constraints.

Pragmatics: Description of context and intention is based either on explicit reference to the enterprise model, to enterprise tasks, to enterprise policy, and environments or on intensional logics used for relating the interpretation and meaning to users depending on time, location, and common sense.

Pragmatism means a practical approach to problems or affairs. According to Webster [908] pragmatism is a ‘balance between principles and practical usage’. Thus, it is a way of considering things. Pragmatism may be based on

methodologies, e.g., database design methodologies. The co-design methodology enables consistent design of structuring, functionality, interactivity and distribution of information systems. The constituents of the methodology are, however, constructs of the design language and constructors used to construct complex construction. Constructs we use are the elements of the extended entity-relationship models [823]: attribute types, entity types, relationship types of arbitrary order, and cluster types. Constructors used for defining more complex types are: (Cartesian) product constructor (for associating types) and list, tree set and bag constructors (for constructing groups or collections of types) [826].

The way of considering things or the practical usage has been investigated in less depth. Schemata have a certain *internal structuring* or *meta-structuring*. Thus, pragmatism of modelling should also be based on principle to handle the meta-structuring within the schemata.

12.4 Bibliographical Remarks

Systematic web development based on modelling techniques has been elaborated in [850, 202, 727]. A *mini-story* [244, 871] typically captures a small, self-contained, tightly connected set of scenes similar to a movie clip. Web engineering is the application of systematic and quantifiable approaches (concepts, methods, techniques, tools) to cost-effective requirements analysis, design, implementation, testing, operation, and maintenance of high-quality web applications [144, 458, 595, 683, 766]. The aim of web application engineering is to present content to a highly heterogeneous audience of web users. Consequently, the emphasis is on content modelling and adaptation, while presentation issues only play a minor role; a general concept for presentation does not exist.

WIS engineering generalises, adapts and revises approaches that have been developed for conceptual modelling of information systems, e.g., [31, 32, 61, 105, 111, 119, 147, 210, 215, 299, 322, 387, 547, 627, 631, 671, 706, 753, 790, 789, 820, 64]. WIS engineering may incorporate techniques for physical design, e.g., [508].

As in daily life, these mini-stories follow general stereotypes that follow the same kind of behaviour. It turns out that mini-stories can be parameterised on the basis of the W*H framework [362, 226, 175]. Typical parameters are: wherefore, whereof, wherewith, worthiness, why, whereto, for when, for what reason, by whom, to whom, whatever, wherein, where, for what, wherefrom, whence, what, how, whereat, whereabouts, whither, when, why, what properties, what scenario, which restrictions.

Database-backed development of information-intensive websites has become nowadays a standard. Early work on this approach and alternative approaches are [144, 158, 251, 253, 269, 268, 282, 292, 518, 563, 686, 690, 762]. WISs require also sophisticated quality maintenance [655].

Key Messages

The Co-Design Methodology for WIS Engineering

- supports requirements capture with respect to business process, application domain, user interfaces and technical infrastructure;
- supports WIS specification concerning both the aspects of the information system and the presentation system;
- has been analysed and assessed to conform to SPICE level 3;
- supports WIS architecture capturing the information view, domain view, data-flow view, deployment view, and infrastructure view;
- addresses all primary dimensions for WIS development known from the Zachman framework and its extensions;
- supports syntactic, semantic and pragmatic aspects of WIS engineering and development.



Systematic Development of Web Information Systems

The co-design framework [823, 833] provides a methodology for description, prescription and specification. A methodology is the study of and knowledge about methods. Methods used in the co-design framework are based

- (a) on the extended entity-relationship model that supports specification of structures and functions,
- (b) on the storyboarding language SiteLang that supports specification of users, their profiles, their portfolio, their actions, and their stories and
- (c) on the framework DistLang that provides a general framework for architecturing, for development of services and exchange frames.

We distinguish a number of facets or views on the application domain. Typical facets to be considered are business procedure and rule facets, intrinsic facets, support technology facets, management and organisation facets, script facets, and human behaviour. These facets are combined into the following aspects that describe different separate concerns:

Structuring of a database application is concerned with representing the database structure and the corresponding static integrity constraints.

Functionality of a database application is specified on the basis of processes and dynamic integrity constraints.

Distribution of information system components is specified through explicit specification of services and exchange frames.

Interactivity is provided by the system on the basis of foreseen stories for a number of envisioned actors and is based on interaction objects which are used to deliver the content of the database to users or to receive new content.

This understanding has led to the **co-design approach** to modelling by specification *structuring*, *functionality*, *distribution*, and *interactivity*.

13.1 Application Domain Description

13.1.1 Application Domain Description and Requirements Statement

The most important outcome of application domain engineering is an application domain model and its associated application domain theory. The main activity is to gather from application domain business users, from literature and from our observations the knowledge about the application domain. It is combined with validation, i.e., the assurance that the application domain description commensurates with how the business users view the application domain. It also includes the application domain analysis, i.e., the study of application domain (rough) statements, the discovery of potential inconsistencies, conflicts and incompleteness with the aim of forming concepts from these statements.

Process Purpose: Goals and Subject

Goals and subject	Application domain description Agreement for development Project scope: Milestones, financial issues Clarification of development goals (intentions, rationale) Sketch of requirements
-------------------	--

Process Outcomes: Work Products as Process Results

The work product is a result or deliverable of the execution of a process and includes services, systems (software and hardware) and processed materials. It has elements that satisfy one or more aspects of a process purpose and may be represented on various media (tangible and intangible).

Documents of the application domain layer are HERM [823] concept maps, HERM functionality feature descriptions, the DistrLang distribution specification resulting in contract sketches which include quality criteria, and the SiteLang interactivity specification of the application story with main application steps.

The documents are combined within the *Stakeholder contract* (“Lastenheft”) and the feasibility study. Additionally a number of internal documents are developed such as life case studies, description of intentions, context specification, and user profiles and portfolio.

	Information analysis missions and goals of the WIS, brand of the WIS general characterisation of tasks and users general characterisation of content and functions description of WIS context Intentions of the web information system, catalogue of requirements scenarios, scenes, actions, context and life cases user profiles and user portfolio actor profiles and actor portfolio, personas Business rule model and storyboards scenarios, scenes, and actions life cases and context WIS utilisation portfolio scenarios, activities supporting content and functions non-functional requirements, context space Metaphor description base metaphors, overlay metaphors metaphor integration and deployment
Developed documents Application domain description section	Stakeholder contract: goal information, concept sketch, product functionality, story space, views on product data, view collaboration sketch Comparison with products of competitors Evaluation of development costs
Developed documents Internal section	Planning of goals, development strategy and plan, quality management Development documents on product components and quality requirements with base practices, generic practices and capabilities, estimation of efforts

Base Activities and Steps

The system development process is usually decomposed into clearly distinguishable development activities that support tackling separate issues at separate times. An activity either starts from scratch and results in a completed work product or starts from a set of work products and results in a completed or revised work product. Activities are work product transformations. Therefore, the steps can be given through refinement activities, i.e., elicitation, determination, instantiation, extension, and fitting. Activities typically proceed in one or (usually) more steps. Activities are describing the ways how the work is performed and can be practiced according to SPICE. A practice contributes to achieving a specific development process purpose or to

the achievement of a specific development process attribute or enhances the capability of a development process. Activities consist of steps.

Let us consider the engineering activities. These activities are enhanced by acquisition activities and activities supporting the life cycle.

We envision three base activities:

1. Development of application domain description
2. Development of the stakeholder contract
3. Development of the internal documents such as description of product components and quality requirements with base activities, generic practices and capabilities and estimation of efforts, etc.

We demonstrate the base activities for the first one. We use the SiteLang specification language [850] and the conceptual framework of [727].

	<ol style="list-style-type: none"> 1. Analyse strategic information <ul style="list-style-type: none"> Specify mission and brand Characterise in general tasks and users Characterise in general content and functions Describe WIS context 2. Derive intentions of the WIS <ul style="list-style-type: none"> Obtain general requirements Extract life cases Describe scenarios, scenes, actions, and context Describe user profiles and user portfolio Derive actor profiles and actor portfolio, personas 3. Extract business rule model and storyboards <ul style="list-style-type: none"> Develop scenarios, scenes, and actions Specify life cases and context Eliciting metaphors 4. Revise business rules of the application possibly with reorganization models 5. Compare new business rules with known visions 6. Compare with products of competitors 7. Derive WIS utilisation <ul style="list-style-type: none"> Describe scenarios to be supported Describe activities based on word fields Describe supporting content Describe supporting functions Describe non-functional requirements Describe the context space
Step 1. Development of the application domain description	

Step 1.	8. Develop metaphor Describe base and overlay metaphors Find metaphor integration Develop templates for deployment
Precondition	Contracted collaboration of all partners Real necessity
Postcondition	Description of application domain accepted and consistent New business rule model accepted

Information analysis is based on WIS storyboard pragmatics. We usually start with templates that specify the brand of the WIS by the four dimensions provider, content, receiver, and main actions. The brand is based on the mission containing general statements on the content, the user, the tasks, purposes, and benefits. The content indication may be extended on the basis of the content item specification frame and by a rough description of context based on the general context template. The brand description is compared with the description of tasks to be supported. Tasks are specified through the general characterisation template. The specification of the intention of the WIS is based on the designation of goals of the WIS.

Additionally we integrate metaphors. They must be captured as early as possible. Metaphors may be based on the style guides that are provided by the customer. Metaphorical structures for WIS can be developed based on linguistic and cognitive research [849].

The application domain description feeds the derivation of the WIS utilisation portfolio. The general section of the WIS utilisation portfolio consists of a list of categories, refined brands, and a general description of kinds of WIS portfolio. The partner section of the WIS utilisation portfolio surveys partners, i.e., a list of actors with portfolio, possibly with rights, obligations, roles. The storyboard section of the WIS utilisation portfolio describes scenarios used and activities which are based on a list of word fields characterising activities. Activities are characterised by an activity style, by activity pattern, and by collaboration styles and pattern. The content section of the WIS utilisation portfolio describes the content chunks together with the content portfolio, i.e., content demand, consumption, and production. The functionality section of the WIS utilisation portfolio describes supporting functions on the basis of function chunks and the functionality portfolio, i.e., demand, consumption, production of functionality. The WIS utilisation portfolio is enhanced by a designation of non-functional requirements, especially quality requirements, and by a derivation of specifics of the context space.

Work Products, Partners, Aspects, and Resources

The work product is a result or deliverable of the execution of a process and includes services, systems (software and hardware) and processed materials. It has elements that satisfy one or more aspects of a process purpose and may be represented on various media (tangible and intangible).

Usable input documents	Analysis of business rules
	Style guides of the customer
	State of application (Hardware, software)
	Product comparison
	Laws, regulations, restrictions
	Templates for application domain description
	Previous WIS developments
Developed documents Results	Metaphor dictionaries
	Information analysis
	Intentions of the WIS
	Business rules and storyboards
	WIS utilisation portfolio
Comparison with products of competitors	

The work products exhibit the description and prescription of the application domain and its phenomena and concepts. Typical phenomena are hierarchical and compositional phenomena, denotable and computable concepts, contextual and state concepts, temporal and spatial phenomena. We may add properties of temporal behaviour such as continuity, discreteness, hybridicity, and chaos, evolution and configuration properties based on statics and dynamics, properties describing the human-computer interaction such as tangibility and intangibility, and finally dimensionality. A phenomenon is humanly tangible if it can be observed by humans. The work products are mainly texts and thus hampered by the one-dimensionality of the textual language. At the same time, applications are highly structured and multidimensional. We can use a plethora of diagrammatic notations. We need however to combine them with formal methods [110, 320].

The partner perspective describes the understanding of the application domain shared by a specifically identified business user group - a view that may differ from one group to another in the same application domain. The believability of the application domain model depends on clear identification and liaising of the model issues. Partners may form a group or team within a collaboration. We distinguish perspectives of business users, of staff supporting the deployment of the WIS, and of stakeholders of the developers' house.

The following table displays the result of this step according to the co-design concerns:

Structuring	Functionality	Distribution & Collaboration	Interactivity	Architecture
business information	business rules	general kinds of collaboration and distribution of business users	general access needs and environment	not considered
Partner	Role			
Owner	placing a request, providing information needs and business rules, agreeing with business rule changes			
Provider	potentially available			
Business user	becoming involved and interviewed			

The role and the involvement of the partner is determined by the concern of the partner. Client executive and other upper-level management expect WIS to improve their competitiveness and position. Tactical and operational management base their expectations on management and organisational issues. Non-management staff expect support in their daily work and to their interfaces with the customer.

We distinguish two extreme groups of partners: Business users and market-driven partners. These two different groups have varying or even contradicting requirements and approaches. Their roles and their participation in the development process differ to a large extent.

Partner	Role
Planner	gathering, interviewing, analyzing, comparing competitors, regards juristical restrictions, proposes quality fulfillment
Owner	placing a request, providing information needs and business rules, agreeing with business rule changes
Financing party	potentially available
Provider	potentially available
Central project manager	becoming informed, develops strategy and plans, derives goals
Designer	starting with ideas, sketches
Developer	provide details on state of application
Business user	becoming involved and interviewed

The four aspects of co-design modelling have both syntactic and semantic elements. They concentrate either on models describing structural properties or evolution of the WIS or the collaboration among actors or the distribution or architecture of the system. [Figure 12.5](#) in Chapter 12 illustrates our considerations.

Resources may be of different kinds. We use resources for enactment of steps. There are mandatory resources such as specification languages and methods. Most of the activities are also based on theoretical fundamentals.

Resources partially contain our knowledge on the theories used within the application domain. We do not need the entire and completely formalised the-

ory of the application domain. Gathering and abstracting the theory may be a very difficult task. We can however start with the topics or annotations that are common in the application domain. Next we gain understanding, discover principles, discuss assertions, and reach some common sense with the stakeholders. Techniques for detection of the theory for the application domain are similar to those we use for modelling, i.e., gaining understanding, obtaining inspiration, presenting and becoming informed, asserting and predicting behaviour, and to capture restrictions of any kind.

Theoretical fundamentals	State-of-the-art Business science Problem solution frameworks User psychology
Methods and heuristics	Interviews Comparison with protocols Observation of similar applications
Specification languages	Natural language Concept and symbol maps Business rules language Application domain description language SiteLang HERM
Infrastructure supporting the step	Brainstorming tools Juristical databases Word processor
Used additional documents	Regulations/laws/restrictions Organization of work Enterprise portfolio Competitors profiles and portfolios

Tips and Tricks

Web information system engineering relies on the experience gained before. So, stereotypes and pattern that have been used before can be reused and correspondingly adapted. Typical reusable decisions are which platform is going to be used, which system environment provides which support, and performance concerns such as update rates.

The *brainstorming* session clarifies which goals and aims are primary, which users or user groups with which information need must be considered, which tasks and which functionality are in the centre of extension, which mini-stories can be typically observed, and which kind of workflow should be developed. The audience of the website results in a weighted list of visitors. For instance, brainstorming for an infotainment site of a city concentrates on: (a) attracting visitors, inhabitants' information, attracting business; (b)

tourists, inhabitants, business people, town officials (and not on kids, pupils, journalists, ...) with information on events, traffic, hotels, restaurants, official town information, education, social life, real estate, ..., doctors and pharmacies; (c) tourist tasks (preparing for visit, reserving and booking, attending), with information on travel schedule, reachability, highlights hopping; and (d) inhabitant needs, e.g., looking for specific town information (health, emergency, ... services, confessional, ... events; spare time activities, culture, sport, small events, clubs; town announcements, traffic, roadwork; restaurants, celebrations; associations, parties, clubs, meetings) or looking for general information such as news, weather, TV, ... program, and actual news.

Based on the brainstorming result we evaluate *goals of the website*. The evaluation is based on the ‘mission’ of the organisation and the kind of website that supports this profile. It is harmonised with the corporate identity of the providers and supporters of the given website. Goals should be ordered by their long-term and short-term importance. The brainstorming might also clarify what will be the website in two years from now. For instance, a book seller website is oriented on selling books. So, it should properly provide information on books. Bestsellers might be offered in third place. The development of a customer community might be more important. In this case, proper reviews would enhance the website. Aims and goals are typically ordered. For instance, the order for business or infotainment sites can be: (1) primary intentions, (2) primary necessities, (3) secondary aims, (4) stimuli, (5) audience education, (6) beauty, and (7) actors’ self-realisation.

So, the first step is based on pragmatism concerns of the website. We clarify in a *usage analysis* therefore (i) life cases (how the stories match the users’ intentions and expectations), (ii) user models (which user/actor profiles and portfolios have to be considered), and (iii) contexts (everything that surrounds and thus impacts on a utilisation situation). This clarification gives us an understanding of the most important part of the WIS portfolio: content and utilisation chunks.

Now we can start with the *analysis of intentions*. Intentions are coarsely formulated as part of a strategic WIS model (mission, purpose, goals). Utilisation scenarios are developed on the basis of intentions. The description of intentions includes from one side a clear understanding of aims and targets of the WIS and from the other side long-range and short-term targets, expected visitors, characteristics of this audience, tasks performed by the users, necessary and nice-to-have content, restrictions of usage, etc. Elementary life cases ruling the visit of a website are briefly sketched. A result of the elementary life case analysis could be the one display in [Figure 5.7](#) which depicts the reasons why somebody wants to visit a certain movie and has the corresponding information demand in dependence on his or her intentions. It allows to derive the service kind, page utilisation, actors involved, presentation pattern, content, and functions. Context (time, regional, content, ...) is integrated into life cases and mini-stories of the application. Life cases may be combined to more complex ones by stepwise infolding progress based on sequential, par-

allel, or concurrent flows of activities. Plots for life cases incorporate with pre-, post-, guarantee- and rely-conditions for each of the steps. Additionally, variations of life cases are considered in dependence on kinds of actors. Life cases satisfy some information demand (consumption) and are supported by data provision, data exchange, and data share. The mini-story description allows to derive characteristics of *data resources*, e.g., (A) collaboration data (user management (roles, profile (esp., polarity and personality), persona), information demand, interactivity support, workspace and work room, communication space, website characterisation data (addressing schemes, access scheme), contacting and responsibilities primarily necessary); (B) profiling of databases (raw data schemata, data provenance, data quality, export/import facilities, data types and mapping, handling missing data, strategies for handling foreign content); (C) data logistics (scheduling, planning of exchange, exception handling, contingency and compensation, delaying, eager management); and (D) data recipes for data representation.

Additionally, *feature and function resources* may be sketched, e.g., (a) integration functions for data from inside/outside; (b) sketches for main features such as search, input, output, history, navigation; (c) sessions, their required form and recipes for session support; (d) approaches for handling horizontal navigation and for integration of other sites; (e) usage habits and resulting combined functions; and (f) function recipes for user requests, presentation style guides.

Guidelines for *content and functionality* depend on (i) analysis of content, navigation, indexing, ... so far; (ii) list of wishes; (iii) derivation of proposals for content; (iv) derivation of functionality necessary; (v) ordering of tasks by priority; (vi) grouping, hierarchies in content; (vii) development of the database schema; (viii) supporting functionality; (ix) platform, systems, distribution; and (ix) update rates.

The *presentation guideline* typically uses a web page as the central exchange medium between system and user. The page design is based on schemes for general placement, parqueting, colouring, texturing, etc. and on support for principles such as principles of visual communication and of visual cognition. We derive therefore applicable styles of consumption (shallow, deep, reasoning during) and use psychogenesis and psychognosis ("Fremdbilder") in order to achieve a good acceptance and reception. This guideline includes styles for composition (e.g., collages, overlays, floating elements, tolerated elements) and identity elements (e.g., logos, icons, backgrounds).

Additional questions concern a rough description of: abilities, knowledge level, information need of users; how to attract surfers; browsing, zapping (if wanted); what interest; changes in information, how to transfer changes to users; do we really need pictures; what is the main statement of the site; how to represent it; who provide the information, data; analysis of competitors for site, audience, utilization; scoop, attraction, news, usability; style of presentation; next update of the site (after analysis of utilization, ...); can we reuse information, data currently available.

It might also be a good idea to consider *business reorganisation* after website launch. This reorganisation includes adaptation and revision of business life after website establishment, transformation strategies for actors currently involved, changes within the organisation (data, function, responsibility, collaboration, exchange reorganisation), reorganisation of side processes, migration strategies for transformation to new business life style, data management as a holistic solution, data stewardship and data quality management, and conformance to laws, regulations, and habits.

A *product, behaviour and competition analysis* is necessary before starting a website project. This analysis includes: state-of-art for website generation tools; (I) analysis of competitor products (walk through techniques, interviews (moderated, informed and skilled users, questionnaires)); (II) run through websites of similar intention; (III) standards and pseudo-standards; (IV) self-descriptions and training guides for the application area, documentation scans; (V) portfolio archeology for market leaders; (VI) gap, SWOT (strengths, weaknesses, opportunities, threats), PURE (positively stated, understood, relevant, ethical), CLEAR (challenging, legal, environmentally sound, agreed, recorded), and SMART (specific, measurable, attainable, realistic, time phased) analyses; and (VII) derivation of test tasks and methods, developing test scenarios, stages (preparation, introduction, test, debriefing), thinking aloud observation, questionnaires and interviews, focus groups, learning from logging actual use, user feedback, choosing test methods.

Finally, metaphors are derived as organisation, functional, and visual metaphors.

13.1.2 Contracting and Documenting

Website development is typically performed by a professional IT institution. Therefore, the financial side cannot be underestimated. So, we should base engineering on a number of contracts.

Step 2.	1. Specification of discourses
	2. Specification of data
Development of the stakeholder contract and of internal documents	3. Specification of functions
	4. Specification of views
	5. Restrictions applicable to product characteristics
	6. Computation of the development costs
	7. Comparison with products of competitors
	8. Milestone sketch, cost frame
	9. Internal documents

These nine documents are the basis for the website development project. This document set can be enhanced by documents or prospects of the partner. One important document is the *presentation style* or the presentation guideline since websites are often judged at the first step by their outside appearance.

The preconditions and postconditions for this step are thus:

Precondition	Availability of information Clarified visions and non-goals
Postcondition	Stakeholder contract developed, consistent and signed Development contract accepted Cost frame fixed and accepted

The stakeholder structure follows approaches used in classical software engineering projects.

Partner	Role
Planer	gathering, interviewing, analyzing, comparing competitors, regards juristical restrictions, proposes quality fulfillment
Owner	placing a request, providing information needs and business rules, agreeing with business rule changes
Financing party	potentially available
Provider	potentially available
Central project manager	becoming informed, develops strategy and plans, derives goals

The co-design concerns are similar to those in the previous step.

Structuring	Functionality	Distribution & Collaboration	Interactivity	Architecture
business information	business rules	general application infrastructure	general access needs and environment	not considered

The basis for this development step is an enhancement of modern software engineering. Since we consider usage-intensive websites we have to use also profiles and portfolio of users. Users have their specific culture, their specific personality (psychogenesis and psychognosis), and their specific habits. So, we have to consider also their expectations.

Theoretical fundamentals	Project planning Work organisation Business science Foundations of software engineering Foundations of business informatics Development psychology State-of-the-art
Methods and heuristics	Interviews Cost estimation methods Comparison with protocols Function point specification Brainstorming on possible development alternatives Risk and cost analysis Introduction of fork points and cutovers

The specification languages for contracting and documenting will be narrative ones in most cases. Competitiveness, argumentation, validation and auditing are improved by inclusion of formal parts into the documents.

	HERM
Specification languages	Natural language
	Project scheduling language
	Stakeholder contract frame

The infrastructure is the same as for other IT engineering projects.

Infrastructure supporting the step	Project planer Word processor
------------------------------------	----------------------------------

A planning step is the formation of a website development team. Each team member has specific competencies and may perform in a variety of roles. There are competencies beyond competencies in classical software engineering projects since website projects use agile development methods and approaches, e.g., communication and acquisition abilities especially with the customer and within the team. Kernel competencies are programming, interface development, storyboard development, and graphics development. Side consultancy is given by psychologists, linguists, and economists. Team members must be able to work at all abstraction levels (template programming, multi-level architectures). Brute-force prototyping can be made with methods developed for agile or extreme programming. The final product should however reprogram prototypes. Team members should have professional skills for database systems, should master at least two programming languages, and should have a good knowledge in communication systems, compiler technology, and operating systems.

A team member may play a number of roles such as web analytic, web and infrastructure developer (including also database development), programmer, tester and assessor, administrator, application expert, and technologist. Specific roles are concerned with acquisition, community maintenance, partnerships, and providers. It is a good idea to have at least one member that is a web visionary. Typically a website team uses outside competence of graphics designers, lawyers' offices, infrastructure providers, and community builders. Each member should develop the product with the consideration of further maintenance, modernisation and system integration. Since websites are aging far faster than typical software and IT products, the last competence is one of the most important.

The development plan covers all eight phases of development. Specifics of each development step are constantly communicated among team members. There is a continuous evaluation and trace of development costs, trace by milestones with reports and fallout plans, and a good documentation of each step.

The additional documents are refinements and modifications of documents that have already developed in the first step. The difference is however that these documents must be now definitive guidelines for further development.

Used additional documents	Regulations/laws/restrictions Competitors profiles and portfolios Team profile and portfolio Analysis of business rules State of application (Hardware, software) Product comparison Catalogue of life case requirements Business rule model Style guides Competitors profiles and portfolios
---------------------------	--

Services Provided by the Website

We may follow classical approaches to development of IT systems. It may, however, be a better idea to develop the website as a suite of services. The service concept has already been discussed in Chapter 11. We may extend this notion by the W*H framework [175]. A website service can be specified as a collection of content offerings supported by annotation facilities and a collection of functions a user might utilise in some life case.

The *involvement* of web parties (i.e., web users or web providers) within service activation is based on the specification of the *role* a party plays during execution of the service, the part the party plays within its portfolio, and the rights and obligations a party has within the given role. The role specifies the behaviour expected from a party. A role is a comprehensive pattern of behaviour and serves as a strategy for coping with recurrent situations and dealing with the roles of others. A role remains relatively stable, even though different parties occupy the position. A party may have a unique style of role execution, but this is exhibited within the boundaries of the expected behaviour of the party. Role expectations include both actions and qualities.

The service specification can be an integral component of the documents developed in this step. A pragmatical approach is the specification of a service following rhetorical frames by answering the the questions *who*, *what*, *when*, *where*, *why*, *in what way*, *by what means*. The service is primarily declared by specifying:

- The **ends or purpose** (wherefore) of the service and thus the benefit a potential user may obtain when using the service. The **purpose description** governs the service. It allows to characterise the service. This characterisation is based on the answers for the following questions: *why*, *whereto*, *for when*, *for which reason*. We call these properties primary since they define in which cases a service has a usefulness, usage, and usability. They define the potential and the capability of the service.
- The **sources (whereof)** of the services with a general description of the environment for the service.

- The supporting means (*wherewith*) which must be known to potential users in the case of utilising the service.
- The surplus value (*worthiness*) a service utilisation might give to the user.

The rhetorical frame can be extended to the questionnaire in [Figure 13.1](#)

Service	Service Name			
Concept	Ends	<i>Wherefore?</i>		
		Purpose	<i>Why?</i>	
			<i>Where to?</i>	
			<i>For When?</i>	
			<i>For Which reason?</i>	
Content	Supporting Means	<i>Wherewith?</i>		
		Application Domain	Application area	<i>Wherein?</i>
			Application case	<i>Wherfrom?</i>
			Problem	<i>For What</i>
			Organizational unit	<i>Where</i>
			Triggering Event	<i>Whence</i>
		IT		<i>What How</i>
Annotation	Source	<i>Where of?</i>		
		Party	Supplier	<i>By whom?</i>
			Consumer	<i>To whom?</i>
			Producer	<i>Whichever?</i>
		Activity	Input	<i>What in?</i>
			Output	<i>What out?</i>
Added Value	Surplus Value	<i>Worthiness?</i>		
		Context	Systems Context	<i>Where at?</i>
			Story Context	<i>Where about?</i>
			Coexistence Context	<i>Wither?</i>
			Time Context	<i>When?</i>

Fig. 13.1. The W*H specification frame for services

This W*H specification frame supports website documentation by providing an informative model of the website for *explanation, informed selection, and appropriation*. It is based on the five-dimensional specification:

- (A) The primary service description describes the service.
- (B) The annotation dimension captures the Party and Activity dimensions.
- (C) The content dimension captures the supporting means by the application domain dimension.
- (D) The concept dimension captures the ends wherefore the purpose of the service answer questions such as *why, whereto, for when, and for which reason*.

- (E) The **added-value dimension** captures the surplus value that defines the usefulness, usage and usability of the services worthiness for systems context, coexistence context, story context and time context.

Tips and Tricks

The specification of user discourses can be based on *mini-stories* as discourse units with exchange protocols, e.g., statement, reaction, feedback, question/answer, information/confirmation, complaint/excuse, non-acceptable sequences. It incorporates the requested services for the given application. It is enhanced by i*-techniques, i.e., soft goals, goals, task, norm, quality constraints, and domain assumptions. Discourse parties are active or passive. A number of conventions is observed for discourses (quality, quantity, relevance, modality). Discourses can be considered to be a composition of mini-stories. Specific composition operations beside the classical process combination operations are in this case connective, co-reference, substitution, ellipse, repetition, lexical association, and comparison.

This approach allows to consider user data as *mini-content* based on a (content, concept, topic)-triple with an integration into the mini-story. Mini-content can thus be combined to content chunks which are received, are of interest after receipt, are processed after recognition, and are integrated by the user after processing. We thus may derive specific content representation forms. A convenient way is to use the meme approach [815]. Content chunks may be enhanced to knowledge chunks that combine content chunks, concept chunks, and topic chunks within a given context.

This approach allows to consider users' functions as a combination of *mini-features* which essentially are combined 'speech-act' functions. Function chunks for data usage are then get/put-functions as composable queries. Function chunks for data compactification/exploration are based on database functionality. Mini-workflows are small local workflow bunches that use mini-play-in/out functions for data. They are supported by persistent functions.

As a combination of this approach, *views* that support users provide (a) mini-content as static, ready-for-flooding, dynamic, foreign content, (b) contingency or compensating content, and (c) integrable concepts as mini-, medi- or maxi-concepts. They can use a flexible ontology or topic annotation depending on user. They are enhanced by escort, side, history or log views enfolded into actual content based on hook and anchor techniques. They can use features for aggregation or disaggregation developed in data warehouse approaches for data exploration (e.g., accordion technique). They can integrate quality management for data that has been delivered in previous story steps.

Restrictions and limitations specify the NONs (not-represented, not integrated, not in this quality, non-functions, nice-to-have ... at later stage, development if budget changes). We have to consider limitations of technology

due to current programming techniques, current databases and software, due to maintenance, and due to budget.

Cost evaluation can be based on the function point method or on balanced scorecards. The internal documentation includes this project calculation. It also includes a project plan with milestones, deliverables, checkpoints, turning points for problematic development stages, responsibilities, capabilities, etc.

13.2 Architecture Design

13.2.1 Architectures and WIS Development

Challenges of Modern WIS

Web information systems (WISs) augment classical information systems by modern Web technologies. They require at the same time a careful development and support for the interaction or story spaces beside the classical support for the working space of users. These dimensions complicate the system development process. Usually, WISs are data-intensive applications which are backed by a database. While the development of information systems is seen as a complex process, *Web information systems engineering* adds additional obstacles to this process because of technical and organizational specifics:

- WISs are open systems from any point of view. For example, the user dimension is a challenge. Although purpose and usage of the system can be formulated in advance, user characteristics cannot be completely predefined. Applications have to be intuitively usable because there cannot be training courses for the users. Non-functional properties of the application like ‘nice looking’ user interfaces are far more important compared with standard business software. WIS-E is not only restricted to enterprises but is also driven by an enthusiastic community fulfilling different goals with different tools.
- WISs are based on Web technologies and standards. Important aspects are only covered by RFCs because of the conception of the Internet. These (quasi-)standards usually reflect the ‘common sense’ only, while important aspects are handled individually.
- Looking at the complete infrastructure, a WIS contains software components with uncontrollable properties like faulty, incomplete, or individualistically implemented Web browsers.
- Base technologies and protocols for the Web were defined more than 10 years ago to fulfill the tasks of the World Wide Web as they had been considered at this time. For example, the HTTP protocol was defined to transfer hypertext documents to enable users to browse the Web. The nature of the Web changed significantly since these days, but there were only minor changes to protocols to keep the Holy Cow of Compatibility alive. Today, HTTP is used as a general purpose transfer protocol which is

used as the backbone for complex interactive applications. Shortcomings like statelessness, loose coupling of client and server, or the restrictions of the request-response communication paradigm are covered by proprietary and heavy-weight frameworks on top of HTTP. Therefore, they are not covered by the standard and handled individually by the framework and the browser, e.g., session management. Small errors may cause unwanted or uncontrollable behavior of the whole application or even security risks.

WIS can be considered from two perspectives: the system perspective and the user perspective. These perspectives are tightly related to each other. We consider the presentation system as an integral part of WIS. It satisfies all user requirements. It is based on real life cases. Software engineering has divided properties into functional and non-functional properties, restrictions and pseudo-properties. This separation can be understood as a separation into essential properties and non-essential ones. If we consider the dichotomy of a WIS then this separation leads to a far more natural separation into information system requirements and presentation systems requirements. The system perspective considers properties such as performance, efficiency, maintainability, portability, and other classical functional and non-functional requirements. Typical presentation system requirements are usability, reliability, and requirements oriented to high quality in use, e.g., effectiveness, productivity, safety, privacy, and satisfaction. Safety and security are also considered to be restrictions since they specify undesired behavior of systems. Pseudo-properties are concerned with technological decisions such as language, middleware, operating system or are imposed by the user environment, the channel to be used, or the variety of client systems.

WIS must provide a sophisticated support for a large variety of users, a large variety of usage stories, and for different (technical) environments. Due to this flexibility the development of WIS differs from the development of information systems by careful elaboration of the application domain, by adaptation to users, stories, environments, etc.

Classical software engineering typically climbs down the system ladder to the implementation layer in order to create a productive system. The usual way in today's WIS development is a manual approach: human modelling experts interpret the specification to enrich and transform it along the system ladder. This way of developing specifications is error-prone: even if the specification on a certain layer is given in a formal language, the modelling expert as a human being will not interpret it in a formal way. Misinterpretations, misunderstandings, and therefore the loss of already specified system properties is the usual business.

Extending WIS Co-Design

We distinguish a number of facets or views on the application domain. Typical facets to be considered are business procedure and rule facets, intrinsic facets,

support technology facets, management and organization facets, script facets, and human behavior. These facets are combined into the following aspects that describe different separate concerns:

- The *structural* aspect deals with the data which is processed by the system. Schemata are developed which express the characteristics of data such as types, classes, or static integrity constraints.
- The *functional* aspect considers functions and processes of the application.
- The *interactivity* aspect describes the handling of the system by the user on the basis of foreseen stories for a number of envisioned actors and is based on media objects which are used to deliver the content of the database to users or to receive new content.
- The *distribution* aspect deals with the integration of different parts of the system which are (physically or logically) distributed by the explicit specification of services and exchange frames.

Each aspect provides different modelling languages which focus on specific needs. While higher layers are usually based on specifications in natural language, lower layers facilitate formally given modelling languages. For example, the classical WIS Co-Design approach uses the Higher-Order Entity Relationship Modelling language for modelling structures, transition systems and Abstract State Machines for modelling functionality, Sitelang for the specification of interactivity, and collaboration frames for expressing distribution. Other languages such as UML may be used depending on the skills of modelers and programmers involved in the development process.

A specification of a WIS consists of a specification for each aspect such that the combination of these specifications (the integrated specification) fulfills the given requirements. Integrated specifications are considered on different levels of abstraction (see [Figure 11.7](#)) while associations between specifications on different levels of abstraction reflect the progress of the development process as well as versions and variations of specifications.

Unfortunately, the given aspects are not orthogonal to each other in a mathematical sense. Different combinations of specifications for structure, functionality, interactivity, and distribution can be used to fulfill given requirements while the definition of the ‘best combination’ relies on non-functional parameters which are only partially given in a formal way. Especially the user perspective of a WIS contributes many informal and vague parameters possibly depending on intuition. For example, ordering an article in an online shop may be modelled as a workflow. Alternatively, the same situation may be modelled by storyboards for the dialog flow emphasizing the interactivity part. This principle of designing complex systems is called *Co-Design*, known from the design process of embedded systems where certain aspects can be realized alternatively in hardware or software (Hardware Software Co-Design). The Co-Design approach for WIS-E developed in the Kiel project group defines the modelling spaces according to this perception.

We can identify two extremes of WIS development. *Turnkey development* is typically started from scratch in a response to a specific development call. *Commercial off-the-shelf development* is based on software and infrastructure whose functionality is decided upon by the makers of the software and the infrastructure rather than by the customers. A number of software engineering models have been proposed in the past: waterfall model, iterative models, rapid prototyping models, etc. The Co-Design approach can be integrated with all these methods.

Architecture Features and Concerns

WIS development results in a number of implemented features and aspects. These features and aspects are typically well-understood since they are similar to classical software products. One dimension that has often been taken into consideration at the intentional level is the level of detail or granularity of the description. Classical databases schemata are, for instance, schemata at the schema level of detail. This schema level is extended by views within the three-level architecture of database systems. These views are typically based on macro-schemata. Online analytical processing and data warehouse applications brought another level of detail and are based on aggregated data. Content management systems are additionally based on annotations of data sets and on concepts that explain these data sets and provide their foundation. Finally, scientific applications require another schema design since they use sensor data which are compacted and coded. These data must be stored together with the ‘normal’ data.

The architectural component has been neglected for most systems since architecture has been assumed to be canonically given. This non-consideration has led to a number of competing architectures for distributed, main-frame or client-server systems. These architectures can however been considered as elements of the architecture solution space.

Therefore the development space for software systems development can be considered to be three-dimensional. [Figure 13.2](#) displays this space.

Web information systems development has sharpened the conflicting goals of system development. We must consider at the same time a bundle of different levels of details, languages and schemata. Systems will not provide all features and aspects to all users. Users will only get those services that are necessary for their work. At the same time, a number of architectural solutions must co-exist.

Development by Separation of Concern

Our approach concentrates on the separation of concern for development. We shall distinguish the user request diploid within a development:

Signatures/ schemata/ specification languages

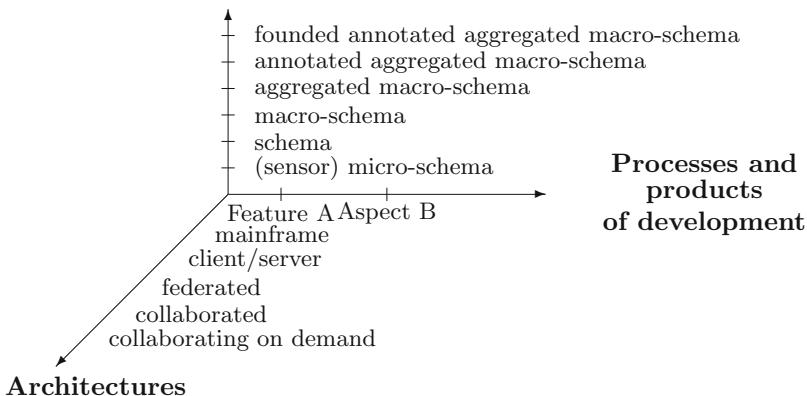


Fig. 13.2. The Development Space for Web Information Systems

Application domain modelling aims in meeting the expectations of the user depending on their profile and their work portfolio. Users want to see a system as companion and do not wish to get another additional education before they can use a system.

Architecture modelling proposes a realisation alternative. This architecture is typically either based on already existing solutions or must be combined with the user system.

Separation of concern for development allows to decompose an application into fields of action, thought or influence. All components have an internal structure formed from a set of smaller interlocking components (sub-component) performing well-defined functions within the overall application domain. Separation of concern covers the what, who, when and (if it is relevant) the why aspects of the business and allows us to identify ‘owners’ and ‘influencers’ of each significant business activity that we need to consult whenever we want to change any of these aspects. A prescriptive (i.e., principles driven) separation is easier to justify to business stakeholders when proposals are put forward to restructure a business activity to improve overall efficiency.

Functional business areas have a high influence on a system. They are identifiable vertical business areas such as finance, sales and marketing, human resources or product manufacturing; and in other cases, they are cross-functional “horizontal” areas such as customer service or business intelligence. Therefore, the business areas already govern the architecture of a system. The establishment of an “ownership” of an information flow assigns the owner to be responsible for making the data available to other business areas as and when those business areas require it. “Influencers” of an information flow need to be consulted when any changes are proposed to ensure that they can comply with the change. Coherence boundaries are the points at which different

functional business areas have to communicate with the outside world in a consistent and grammatically structured language.

13.2.2 Architecture-Driven Engineering

Most approaches known so far did not take into consideration architectural issues. Typically, they are taken for granted or assumed on default. Architectures have a deep impact on the development methodology. Architecture-oriented development is thus a central concern for WIS engineering. The choice of an appropriate architecture is difficult because of the large variety of systems that are going to be considered, e.g., the supporting systems, the content management systems that are used for content delivery and integration, the user management system, the website playout system, and the variety of browsers that might be used by users.

Specification of architecture schemes	
Goals and sub-project	application architecture view
	infrastructure architecture view
	component architecture sketch

One dimension of software engineering that has not been yet integrated well is the *software architecture*. Modelling has different targets and quality demands depending on the architecture. For instance, mainframe-oriented modelling concentrates on the development of a monolithic schema with a support by view schemata for different aspects of the application. Three-tier architectures separate the system schema into presentation schemata, business process schemata and supporting database schemata based on separation of concern and information hiding. Component architectures are based on ‘meta-schemata’ that describe the intention of the component, the interfaces provided by the component, and the bindings among the interfaces. SOA architectures encapsulate functionality and structuring into services and use orchestration for realisation of business tasks through mediators. Therefore, application domain description is going to be extended by consideration of architectures and environments. Software architecture is often considered from the technical or structural point of view and shows the association of modules or packages of software. Beside this structural point of view we consider the application architecture that illustrates the structure of the software from the application domain perspective. Additionally we might include the perspective of the technical infrastructure, e.g., periphery of the system. These three viewpoints are one the most important viewpoints of the same architecture. We call an architecture documentation *architecture blueprint*.

	Architecture schemes application architecture technical architecture technical infrastructure architecture architecture blueprint
Developed documents	Integration schemes and integration obligations WIS development architecture guide View collaboration associations

Classical approaches consider the three facets of system development: application domain description, requirements prescription and software specification. We discover in this paper that there is a fourth facet that cannot be neglected: architecture of the system. Therefore, we extend the classical framework to the *software modelling quadruple* in [Figure 13.3](#).

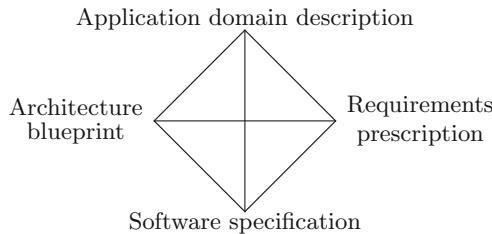


Fig. 13.3. The Software Engineering Quadruple

Software engineering has divided properties into functional and non-functional properties, restrictions and pseudo-properties. This separation can be understood as a separation into essential properties and non-essential ones. If we consider the dichotomy of a WIS then this separation leads to a far more natural separation into information system requirements and presentation systems requirements. The system perspective considers properties such as performance, efficiency, maintainability, portability, and other classical functional requirements. Typical presentation system requirements are usability, reliability, and requirements oriented to high quality in use, e.g., effectiveness, productivity, safety, privacy, and satisfaction. Safety and security are also considered to be restrictions since they specify undesired behaviour of systems. Pseudo-properties are concerned with technological decisions such as language, middleware, operating system or are imposed by the user environment, the channel to be used, or the variety of client systems.

It seems to be most appropriate to use the QuASAR approach to architecture development. In the QuASAR framework the specification can be based on some system architecture, e.g., on a separation into the application interface, a technical system, and the technical infrastructure. The system should reflect all needs of the user.

As we orient our development on a component approach, we use an explicit integration specification for the subsystems. The co-design approach uses the 3C framework to collaboration. We thus choose the corresponding collaboration pattern and styles. The category of a WIS allows us to determine which architecture style is the most appropriate. For instance, infotainment, identity, and community sites are well supported by the PCMEF architecture (presentation-control-mediator-entity-foundation). Information-intensive WISs are typically enhancing content management architectures by the data warehouse paradigm, e.g., in a 3-tier database-backed system. Service-intensive e-businesss WIS might follow a web services architecture, e.g., in the case of online shops. Edutainment systems that go beyond lecture slide delivery might combine data warehouse approaches with those for community websites.

The data warehouse approach is well-supported by interaction schemata. We separate input and output data, use mediators for the association between providers and information users, differentiate the functionality, support security and privacy concerns, may provide an input independence for different channels, can support different output formats (internet, cable net, TV, etc.), and can integrate enterprise networks with intranets and extranets. The disadvantages of such solutions are performance issues due to redundancy and data maintenance, storage requirements, and the necessity for complete acquisition of all potential user requests in advance. The main advantage of this decision is however the direct information gathering and delivery based on the stories and scenarios.

This development step thus clarifies a number of issues

- for system construction (components, interfaces, services),
- includes the meta-structure of the WIS,
- provides viewpoints that depend on the WIS purpose,
- conceptualises components, obligations, permissions,
- defines the abstraction layers used for WIS development
- uses architectural pattern, and
- gives an insight into the quality of service as one of the assessment criteria.

Step 3. Development of WIS architecture	<ol style="list-style-type: none"> 1. Development of the application architecture 2. Development of the technical architecture 3. Development of the architecture of technical infrastructure 4. Development of integration schemes and derivation of integration obligations, further view collaboration associations 5. Derive work partitioning
--	---

Precondition	Stakeholder contract entirely accepted Organization models completely specified
Postcondition	Saturated component architecture

The co-design concerns are now concentrated around the specification of the distribution. The architecture of a system is ruled by the decisions for distribution. The simplest architecture for continuously evolving WIS is the component-oriented architecture with some infrastructure systems, e.g., the architecture in [Figure 11.9](#).

Stakeholders during this development step are almost the same as in the previous steps. Since a number of decisions are made with the settlement of an appropriate architecture, the architecture must be coherent with the entire environment of the providers and supporters.

Structuring	Functionality	Distribution & Collaboration	Interactivity	Architecture
business information	business rules	general application infrastructure, general kinds of collaboration and distribution of business users	general access needs and environment	now sketched

Distributed WISs also support changes made during development and later deployment or maintenance. In this case, we may support *component independence* similar to logical or physical independence of database systems. Component-based systems allow flexible refinement by instantiation, specialisation of context extension. They use pattern for the construction in the large. Systematic quality control can be delegated to quality control for the interface of the central infrastructure system and to quality control of the component systems.

In this step the roles played by the partners change to:

Partner	Role
Planner	gathering, interviewing, analyzing, comparing competitors, regards juristic restrictions, proposes quality fulfillment
Owner	placing a request, providing information needs and business rules, agreeing with business rule changes
Financing party	potentially available
Provider	potentially available
Analyst	developing potential architectures
Security manager	developing security and privacy schemes
Central project manager	becoming informed, develops strategy and plans, derives goals
Designer	starting with ideas, sketches
Developer	provide details on state of application
Database programmer	check possible architectures
Business user	becoming involved and interviewed

Finally we arrive with a number of documents. Agile or extreme development methodologies avoid documentation in such early stages and rely on responsibilities of developers. Since WIS development must also result in a manageable system we prefer early and high quality documentation.

Usable input documents	<ul style="list-style-type: none"> Stakeholder contract Description of enterprise profiles and of business processes Organization models
------------------------	---

The fundamentals, theories, and languages are similar to those in the previous steps. Component architectures are the main addition we have to consider. During this step we also decide the configuration of the trinity: database management systems to be used, web server, and web application server. Technologies are not equivalent from the perspective of soundness, usability, productivity and so on. Each of these systems have their advantages and disadvantages. So, we have to compromise among alternatives. This compromise rules also the decision which client-server techniques are the most appropriate, e.g., servlet, applet, JSP, JMS, etc. The last decision also depends on the kind of requests issued by the user to the system, e.g., static or dynamic, eager or moderate,

Theoretical fundamentals	<ul style="list-style-type: none"> Construction science Foundations of business informatics Component theories Software architecture
Methods and heuristics	<ul style="list-style-type: none"> Mappings of general relationships within the organization
Specification languages	<ul style="list-style-type: none"> HERM Architecture description language Integration and collaboration frame language
Infrastructure supporting the step	<ul style="list-style-type: none"> Graphics tools Word processor

Used additional documents	Established architectures Regulations/laws/restrictions
---------------------------	--

The purposes of explicit architecture specification are *construction of systems, communication among all stakeholders, analysis of the system, examination and check, documentation, mastering of application complexity, and support for continuous improvement.*

13.3 Requirements Analysis

13.3.1 WIS Requirements Analysis

The purpose of the requirements analysis process is to establish the requirements of the software components of the system. As a result of successful implementation of the process we look for satisfaction of a set of properties such as:

- the requirements allocated to software components of the system and their interfaces will be defined to match the customer's stated needs;
- analyzed, correct, and testable software requirements will be developed;
- the impact of software requirements on the operating environment will be understood;
- a software release strategy will be developed that defines the priority for implementing software requirements;
- the software requirements will be approved and updated as needed;
- consistency will be established between system requirements and design and software requirements;
- the software requirements will be communicated to all affected parties.

The verb require means 'to seek for'. Therefore, we need to answer the basic question *Who needs what, why do they need it?*

For WIS, requirement analysis is enhanced by content chunks (representing content, concepts and topics), function chunks (representing all functionality requirements), storyboard of the WIS, interaction type requirements (data, content, functions, containers, performance, operational, maintainability, support, security), representation requirements (e.g., look and feel, grids and pattern), and general storyboard requirements (e.g., usability, humanity, privacy, cultural, political, legal).

Goals and subject	Requirements elaboration Requirements analysis WIS documentation "Pflichtenheft" Development contract
-------------------	---

We may use the SPICE framework for WIS requirements analysis. The SPICE ENG 1.2 framework uses the following base practices which are also relevant for WIS development:

ENG.1.2.BP1: Specify software requirements. Determine and analyze requirements of the software components of the system and document in a software requirements specification.

ENG.1.2.BP2: Determine operating environment impact. Determine the interfaces between the software requirements and other components of the operating environment, and the impact that the requirements will have.

NOTE: The operating environment includes tasks performed by, or other systems used by, the intended users of the software product.

ENG.1.2.BP3: Evaluate and validate requirements with customer. Communicate the software requirements to the customer, and based on what is learned through this communication, revise if necessary.

ENG.1.2.BP4: Develop validation criteria for software. Use the software requirements to define the validation criteria for the software. The validation criteria are used in developing the software tests.

ENG.1.2.BP5: Develop release strategy. Prioritize the software requirements and map them to future releases of the software.

ENG.1.2.BP6: Update requirements. After completing an iteration of requirements, design, code, and test, use the feedback obtained from operation to modify the requirements for the next iteration.

ENG.1.2.BP7: Communicate software requirements. Establish communication mechanisms for dissemination of software requirements, and updates to requirements to all parties who will be using them.

ENG.1.2.BP8: Evaluate the software requirements. Evaluate the consistency and establish traceability between software requirements and system requirements.

A classical approach in the waterfall development model uses a complete and detailed description of all requirements. This WIS development and system documentation (called in German "Pflichtenheft") describes the WIS as a product via its data, its functions, its performance and quality parameters, its functional and non-functional properties, its testing environment, its interfaces, its required infrastructure and environment, its conditions for deployment, its user support, etc. There are many standards, e.g., IEEE Recommended Practice for Software Requirements Specifications; IEEE Recommended Practice for Software Acquisition; IEEE Standard for Software Safety Plans; IEEE Guide for Developing System Requirements Specifications; IEEE Standard for Functional Modeling Language – Syntax and Semantics for IDEF0; IEEE Standard for Conceptual Modeling Language Syntax and Semantics for IDEF1X97 (IDEFobject); IEEE Guide for Information Technology – System Definition – Concept of Operations (ConOps) Document; IEEE Standard – Adoption of International Standard ISO/IEC 12119: 1994(E) – Information Technology – Software packages – Quality requirements and testing; IEEE Adoption of ISO/IEC 14143-1:1998 Information Technology – Software Measurement – Functional Size Measurement – Part 1: Definition of Concepts.

The WIS development and system documentation can also be used as a basis for the development contract of a website company. So, the goals, the subject, and the developed documents consist of:

Developed documents	<ul style="list-style-type: none"> WIS utilisation portfolio <ul style="list-style-type: none"> scenarios, activities supporting content and functions Requirements portfolio <ul style="list-style-type: none"> content and function requirements interaction type requirements storyboard requirements representation requirements non-functional requirements, context space WIS documentation <ul style="list-style-type: none"> "Pflichtenheft" WIS development and system documentation Security and privacy view architecture
Potentially revised documents	<ul style="list-style-type: none"> Refinement of architectures Stakeholder contract Utilization scenarios

WIS requirements analysis may start with a number of *activity models*. As examples of activity models we may use an extension of brand and mission description of website, e.g.,:

Business models reflect the method of doing business by which a company can sustain itself - that is, generate revenue. Internet commerce will give rise to new kinds of business models. The web is also likely to reinvent tried-and-true models. Basic categories of business models on the Web are brokerage; advertising; infomediary; merchant; (direct) manufacturer; affiliate; community; subscription; and utility.

These categories are vantage points for the mini-story specification. Community models follow the typical and well-established ways how communities act. We may use the W*H framework for community modelling. The viability of the community model is based on user loyalty. Revenue can be based on the sale of ancillary products and services or voluntary contributions.

Subscription models describe the way how users utilise a website. For instance, users are charged a periodic – daily, monthly or annual – fee to subscribe to a service. Subscription fees are incurred irrespective of actual usage rates.

Utility models or “on-demand” models are based on metering usage or a “pay as you go” approach. Unlike subscriber services, metered services are based on actual usage rates.

These models enhance the features introduced in the previous chapters. So, a requirements specification step might consist of the following activities:

Step 4. Derivation of WIS requirements	<ol style="list-style-type: none"> 1. Derive WIS utilisation <ul style="list-style-type: none"> Describe scenarios to be supported Describe activities based on word fields Describe supporting content Describe supporting functions Describe non-functional requirements Describe the context space 2. Derive mandated constraints <ul style="list-style-type: none"> Describe solution constraints Derive WIS implementation environment Specify partner applications and off-shelf software Derive anticipated workplace environment Revise schedule and budget constraints 3. Derive security and privacy views 4. Derive requirements portfolio <ul style="list-style-type: none"> content and function requirements media type requirements storyboard requirements representation requirements non-functional requirements, context space 5. Agree on naming conventions and definitions <ul style="list-style-type: none"> glossary 6. Combine all documents into the WIS documentation
Precondition	<ul style="list-style-type: none"> Clarity on life cases Agreed architecture Organization model known & stable
Postcondition	<ul style="list-style-type: none"> WIS development and system documentation Obligations for requirements prescription layer

Therefore, classical requirements determination is enhanced for WIS engineering by a storyboard through application scenarios, by context that might be flexible and negotiable, by intention specification that is now explicit, by presentation guidelines, and by an integrated system view within the co-design approach.

The WIS development and system documentation may be guided by primary concerns that govern other concerns. As already discussed, the intention and user specification is a starting point for all categories of WIS. Business WISs are typically oriented on the business story and the business context. Community WISs are oriented on the community story and the requested functionality (and maybe content). Edutainment WISs are governed by their story space and the content. Infotainment WISs are oriented on the content. Entertainment WISs which are typically not supported as information-intensive sites follow a story-function-presentation guideline. Identity websites are mainly oriented on the identity that should be presented.

The application domain description can be used to deduct the *requirements prescription* on the basis of *functional requirements*, through *non-functional requirements*, and additionally through the *Storyboard* of the application. So, the development and system documentation may be structured as follows:

WIS Development and System Documentation	Goal determination Description of structuring Description of functionality Description of utilization scenarios and activities Description of views and distribution Actor model, task skeletons
---	---

This list is similar to the items in software engineering and does not need further explanation.

Add-On's to the WIS Development and System Documentation	Presentation style, layout Quality goals, global test scenarios, test cases Development environment Installation conditions user instruction, maintenance problems, norms, regulations, patents, licences and users' handbook (with application scenarios and application samples)
---	---

The presentation guide is an essential element of any website development. Web pages are at the first step judged by their presentation. It must be pleasing and follow the principles of visual communication visual cognition, and visual design. The list of quality goals is often rather long. A trick is in this case to weight the goals and then to classify them into must-satisfy, good-to-satisfy, nice-if-satisfied, and optional goals.

The technical team has already to decide at this stage all main concerns for implementation. They must also consider actual regulations such as copyrights, etc.

Appendix for the WIS development and System Documentation	Definitions of notations by a glossary or by a notation lexicon System acceptance guideline social acceptability practical acceptability usefulness utility, usability cost, compatibility, reliability
--	--

The *glossary* is a list of terms with the accompanying definitions. All terms must be properly explained. It typically goes beyond lexicography or lexicology. Sometimes glossaries are informally represented by an ontology.

WIS acceptance is a main concern already at this phase. *Social acceptability* refers to the ability to accept - or to be able to tolerate - differences and diversity brought into the community of WIS users by the WIS. It is the basis for positive and lasting deployment of the system. Social resistance on emotional, cognitive, behavioural, and biological responses against a website is a killer argument. Social acceptability is culture-dependent. Therefore, we must evaluate the requirements against the acceptance and cultures of the community of users. For this reason, contributions made by psychologists are essential. Social acceptance requires also proper support for security and privacy.

The triangle (or 3 U's) usefulness-usability-usage is a main concern for any website development.

Usability: The international standard, ISO 9241-11, provides guidance on usability and defines it as: "*The extent to which a product can be used by specified users to*

achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.”

Usability is a mesh between people, life cases and technology. It is about:

- Effectiveness - can users complete tasks, achieve goals with the product, i.e., do what they want to do?
- Efficiency - how much effort do users require to do this? (Often measured in time)
- Satisfaction - what do users think about the products ease of use?

which are affected by:

- The users - who is using the product? E.g. are they highly trained and experienced users, or novices?
- Their goals - what are the users trying to do with the product - does it support what they want to do with it?
- The usage situation (or ‘context of use’) - where and how is the product being used?

Usage expresses flexibility, adaptability and suitability. It is based on

- the act of using something,
- the way that something is used, and
- the amount of something that is used.

Usefulness: concentrates on

- the quality of having utility and especially practical worth or applicability.

We thus derive the 3U-requirement: adaptable and easy to use systems without any additional learning effort. Website development approaches used to call such systems '*grandmother-proof*' or with ease of use: usable without additional training, simple operating, obvious operating, simple for everybody, straightforward, within expectation, with context-sensitive help, with adaptable selection of wording, and with simple dialogues. Additionally, smart systems must be of high utility, i.e., simple to remember, error robust, reliable, of high quality. The *HOME* acronym combines the four quality characteristics of systems: *H*igh quality content, *O*ften updated, *M*inimal effort (e.g., download and processing time and space), and *E*ase of use.

A trick is the task specification that is *usage-centred*. Remember, a description of tasks includes: identifying essential user and business purposes, understanding targeted users by roles in relation to site, understanding tasks in terms of user intentions and needs, prioritising user roles and user tasks in terms of expected frequency and business importance, and engineering the design of a technical system to fit user and business priorities.

Applications are becoming *smart* if they are *Simple* in any step of usage that any user might request, are *Motivational* for any user independently of the way of working of the given user, are *Attainable* for the goals the user has in mind, tend to be *Rewarding* since they seem to be worthwhile, rightly timed, match efforts and needs, and are *Time-efficient* within the limits and expectations of users.

The co-design framework is now oriented on the following issues:

Structuring	Functionality	Distribution & Collaboration	Interactivity	Architecture
business information	business rules	application infrastructure, collaboration and distribution	general access needs and environment	fixed now

Within these concerns, we derive the content schemata (database support, views, interaction types, etc.) and the functionality schemata (especially search, navigation, import, export, operations for interaction types and the interaction schema, etc.). These schema must be adaptable and provide support for a variety of viewpoints. Adaptability must become automatic for runtime adaptation to the user environment and to the network. We also need support for different levels of granularity thus providing a generic way to include different versions and to extend these similar to hierarchies in OLAP systems with `drill-down_roll-up_slice_dice_rotate` functionality.

The context specification is an essential element within this step. We consider the various kinds of context according to Sections 1 and 5.4. The context is directly linked to the interaction types and the interaction schema.

In a similar form, presentation options and grids are specified as a presentation model on the basis of presentation grids. Since the browsers may have their specifics and users may use pages in different modes (e.g., half-screen mode) we derive a guideline for automatic adaptation.

In a similar form we can describe the involvement of different stakeholders during requirements analysis:

Partner	Role
Planner	gathering, interviewing, analyzing, comparing competitors, regards juristic restrictions, proposes quality fulfillment
Owner	placing a request, providing information needs and business rules, agreeing with business rule changes
Financing p. Provider	involved into contracting
Security man. Central project manager	placing requests and restrictions developing security and privacy schemes becoming informed, develops strategy and plans, derives goals
Designer	starting with ideas, sketches
Analyst	starting with ideas, sketches
Developer	provide details on state of application
Story dramaturgist	check possible storyboards
Story scenographer	check possible layouts
Database program- mer	check implementability
Business user	becoming involved and interviewed

The WIS requirement analysis is based on documents that have been developed in the previous steps:

Usable input documents	Stakeholder contract
	Development methodology, planning
	Life cases, story specification, activity documentation
	Architectures
	Style guides (presentation specification)
Theoretical fundamentals	Organization model
	Methodology of storyboarding
	SiteLang theory and theory of media types
	Theory of parallel processes and Petri nets
	Theory of abstract state machines
	Object-relational view theory
	Dramaturgy and scenography
	Psychology of collaboration and cognition

Methods and heuristics	Elicitation of application steps
	Interview techniques
	Observation techniques, preferences
	Competitors products
	Menu, frame and presentation techniques

There are also usability heuristics that can be incorporated into the documents. They are also the basis for the website evaluation.

1. Simple and natural dialogue, optimizing user's operations: Each interaction object has to be well understood, learned. Any information must be provided at the right moment and in the right dose. Adhesion of information must be correct and provide an explicit (de)composition of content. Graphic design and coloring must be consistent (based on the 'gestalt' principles (closeness, closure and similarity)). Typically, less is more. Information is decomposed into foreground information and background information. Sequencing, clustering, and emphasis should be adaptable.
2. Speaking in the user's language (not the designer's): Web pages should be without additional coding and should follow naming conventions. i.e., speaking in the mother tongue with a possibility to use the most appropriate user vocabulary. The language is user-oriented (e.g., "you bought" instead of "we sold you"). User-based metaphors meet their expectations while observing local differences (nation, nationality, group, ...).
3. Minimizing user memory load, cognitive directness: Pages should provide a context-sensitive help (e.g., data format), should be based on generic functions and should support compactness and branching features.
4. Consistency: Similar actions should have with a similar effect. Same information must be given on the same location. Common standards must be observed. User's expectations should be met.
5. Feedback: Usability also requires permanent context-sensitive escort feedback, and error-warning (ahead). Information on response time is necessary (0.1 sec (no), 1 sec, 10 sec (mandatory)). Any system failure, communication failure, or slowdown of throughput should be reported.
6. Clearly marked exits, additional exits: Users control their actions and dialogues, and the undo/redo.
7. Shortcuts: We need a specific navigation for experienced users, back/forward features without using the browser features, typing/clicking ahead features (typing 'B' as a location can directly be extended to Berlin), simple names and labeling, commands instead of actions, and reuse facilities based on previous interaction history instead of redo. The integration of spelling corrections (e.g., algorithms like SoundEx) is a really nice feature.
8. Good error messages (for all six dimensions): Errors must be reported in clear language (without MS code), precise, with reasons, polite (no blame), and by simple messages instead of multiple ones. Users need a clear understanding of the system. Feedback is actor-dependent and sometimes even user-dependent. Messages must include constructive help for solution and support simple recovery.
9. Prevent errors: It is a good idea to support recovery from typing errors and to re-confirm dangerous or potentially harmful actions. Websites might hide system modes or if necessary then show modes with well-designed explanations.

Similarity for different actions should be avoided. Errors should be internally documented for next versions of the WIS. The integration of spelling corrections (e.g., algorithms like SoundEx) is a really nice feature.

10. Context-sensitive online help and documentation: Users never or seldom read documentations. They test instead. Therefore, we should develop help scenarios for main actions (search, understand, apply) and develop synonym networks for better usage. It is a good idea to use varying granularity (from minimal to full information) and page survey cards.
11. Heuristic evaluation guide: For later evaluation and assessment, we use a guideline for systematic evaluation of all scenarios and actions that incorporates also full written reports, statistical tests, independence of test groups, reliability tests, performance tests, and conformance and adequacy of own standards (against (wrong) industrial standards).

These heuristics have already been developed in the 1980s but are equally important nowadays.

Specification languages	HERM rough schema HERM view sketches HERM or ASM business processes SiteLang, media types and DistrLang WIS specification and documentation frames
Infrastructure supporting the step	Storyboard editor Database design editor Word processor
Used additional documents	Regulations/laws/restrictions Similar WIS documentations Trend studies

Collaboration and Complex Storyboards

Storyboards may be extended to *collaboration storyboards*. We represent a combined storyboard for three actors in [Figure 13.4](#). Collaboration storyboards are organised bundles of stories.

- We introduce different types of actors and represent their activities together with the interaction types that support their collaboration. The interaction types can be bilateral (e.g., wiki delivery box) or used by groups or the world (e.g., group communication content chunk space; group help hints and tricks room).
- A story might have a history (the H-circle) for resuming the activity that has been used most lately.
- Actors may use different roles at the same time. E.g. they might be a wiki team member and a supporting expert that provides tips and tricks.

A *wiki* is a collaborative web site set up to allow user editing and adding of content by any user who has access to it. Wikis have changed access and habits of internet users. The main functionality provided for wikis is:

- management of content chunks with their structure, intention, origin and topic,
- annotation management for users depending on their role, rights and obligations,

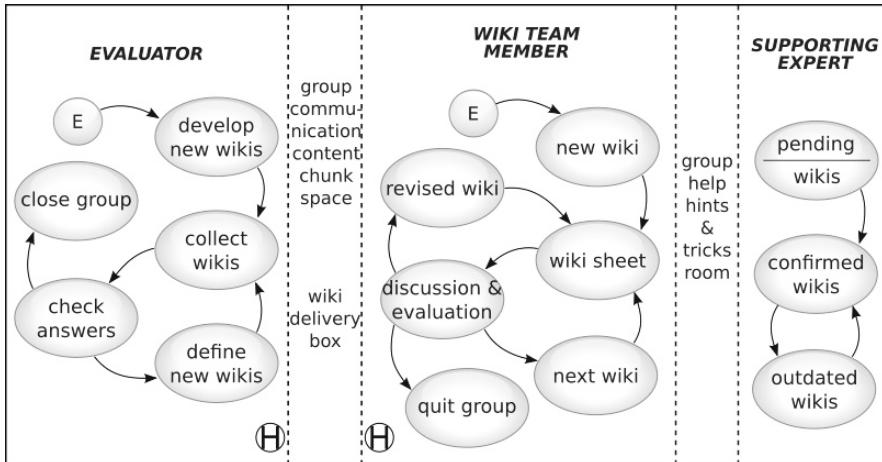


Fig. 13.4. A wiki storyboard within a collaborative learning application

- explanation, exploration and knowledge elicitation and gathering support for readers of content chunks,
- presentation of information in a variety of ways and for a variety of environments,
- user management depending on the roles, functions and rights a user may have on the content,
- security and safety management for integrity of content and information, and
- history and evolution management that allows to show the development of the wiki and to restore previous versions.

Another extension of the storyboarding approach that is useful is the separation of concern within a story and the separation into side and main stories. Let us use the community example in [Figure 13.5](#).

This sample story space for a paper submission and reviewing system pictures the stories around a program committee session where a program committee discusses in an online session the acceptance of papers. The story space is separated into a main story space and the side stories that can be used in parallel. The main story can run in the main browser window and the other side stories can be called on demand. The later ones allow also bilateral communication beside the main session.

13.3.2 Companion Activities at Requirements Prescription Layer

The requirements prescription layer may be enhanced by the development of *genre guides*. These guides support reuse of documents for other projects and nicely guide the current project. They describe the main elements of the interaction schema, of the functionality, of actor specification, metaphors, infrastructure solutions, etc. in a generic form. It is infeasible to build a website on its own and from scratch due to the variety in applications, the velocity of changes, the variations we would need, and the different viewpoints. So, a better way is to model this adaptability at the conceptual level and to envision it at the requirements analysis level. We thus need

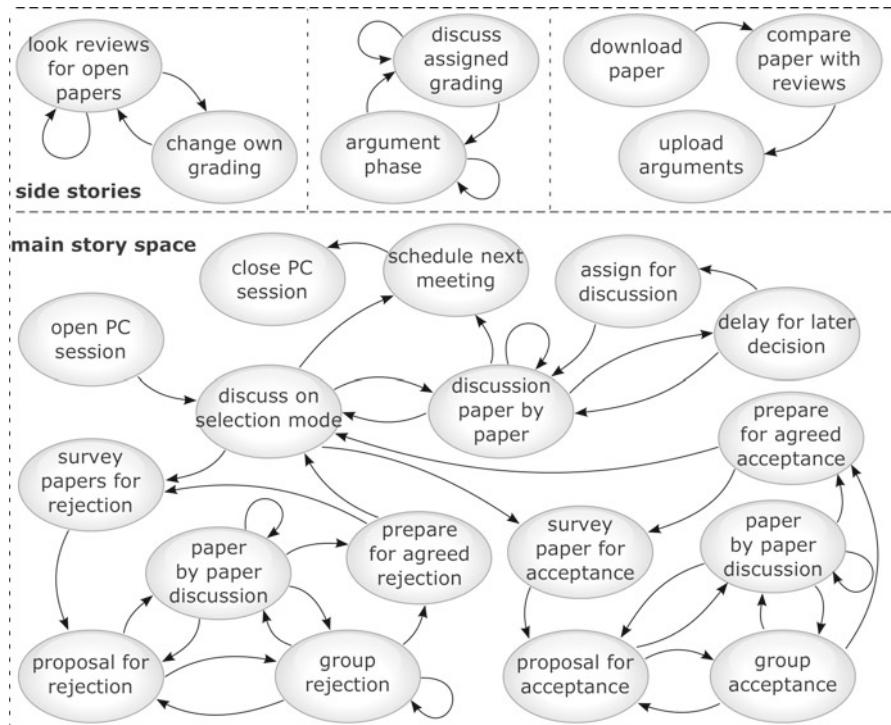


Fig. 13.5. The separation of a storyboard into main stories and side stories

a general model of such systems which can be adapted to users. If we know the main general behavioural pattern of users and some context then we can adapt systems on a basis of a *generic model* with some *adaptation model*. The adaptation model is a *governing strategic and tactical model*.

Genre stereotypes help to understand common patterns typical within a category of websites or more specifically within a sub-category - called genre - of a category. Each kind of genre brings in its specific stereotype. Stereotypes should satisfy at least five properties: (1) they must be accurate; (2) the quality of the stereotype allows it to be used consciously; (3) they should be descriptive, not evaluative; (4) they should be flexible so that they can be modified from time to time; (5) they can be used as a first “best guess”.

These genre stereotypes may be enhanced by actor or user stereotypes, story stereotypes, context stereotypes, content stereotypes, functionality stereotypes, and presentation stereotypes. For instance, a *user stereotype* is given by three dimensions: (1) personae, (2) user model, and (3) user portfolio. A typical stereotype is the *functionality stereotype*. It describes the ways to navigate, provides a link and search guide, characterises organisation, sketches the export/import guide, and proposes approaches to repetition or replication and redundancy. The *presentation guide* is based on presentation pattern for various kinds of interaction objects and proposes specific presentation elements (texture and content; sidebar guides, screen-size

guides; functionality structuring; site-leveling of presentation). The stereotypes may also be combined with an *implementation stereotype* and *implementation pattern*. They include adaptation guides, access guides and context guides.

13.4 Presentation System Specification

Webpage and website design is still more an art than a technology. The presentation system specification includes the graphical design. It is based on the technology of browsers. It uses the frameworks that have been developed for graphical user interfaces. Presentation includes however also development of the story space in such a way that users are supported

- by best-suited content,
- in an understandable presentation and in an easy to survey form,
- at the right amount,
- just at the right moment of time and in time,
- just for current payment, and
- to the right user environment and to the right device

So, the presentation system is user-oriented, is adaptable to any kind of user context, uses high-quality and evaluated content, is situation-related and uses a smart presentation.

Presentation system specification is based on the entire material provided in previous Chapters 1-12. We do not repeat all elements of these chapters but concentrate on additional insights in this subchapter. The presentation system is mainly the specification of the left side in [Figure 11.6](#). It thus describes the socio-technical system in full detail.

Goals and subject	Specification of utilization scenarios (treatment) of the story space Specification of the storyboard Portfolio and generical profiles of all actors Prescription of media types and containers Finalized treatment and high-level blueprints of the story space Specification of the playout system Specification of the layout system based on graphics and generation engines
Developed documents	Life cases of potential users with their full social embedding Decoupage of the story space and high-level blueprints Presentation grids Actor model : detailed portfolio and profile with their cultural stereotype Collaboration frames, e.g., view collaboration, exchange frames Overlay views defined over the ground schema Support frames for dialogue functions Obligations for interaction types Schemata to be used and external schemata Integrity constraint enforcement frames Identification frames based on identification pattern Business processes frames

Potentially revised documents	WIS development and system documentation Utilization scenarios of the story space Database schema and Views Obligations for functionality development Activities at the business user layer Obligations for schemata, views, story space Story space specification Obligations for conceptual WIS development
-------------------------------	--

The treatment with the corresponding utilization scenarios depends on

- information need, content of interaction objects,
- utilization pattern, main navigation traces, and
- user specific utilization and context.

We already discussed the trick to use personas as prototypes (person with a name, profession, aims and goals, technical equipment, behavior, skills and profile, disabilities (e.g., motor, visual, cognitive, spelling), hobbies, habits). For instance, an infotainment site should attract visitors, should provide inhabitants information, and should be informative. So, we can refine the presentation design based on following elements:

Utilisation scenario: explanation, informed selection, appropriation; inform the proactive information seeker according to *information demand, profile, and portfolio*; steering and directing

Methods provided with the site: communication, orientation, combination, survey, feedback

Cargo of the site: *mission* in the utilisation scenarios, *determination, abstract declaration* of the meaning, narrative explanation of the *identity*

Content of the site: main ingredients of the site, how to use as explanatory statement extended by

- *how* the website is used,
- what is the *main background* behind this site, and
- *why* to use this site.

This frame follows the SMART frame. We combine the SMART frame with *persona description*. Remember the persona of Hans Dampf (Jack-of-all-trades) (business man, short-term oriented, interested in culture and history, short distances, hotels at least 4-star, in hurry, good dining and talking, familiar with technology). The scenario used in this case would be based on the specific goals and information needs of this persona ((visits information site for business trip preparation; hotel, spare time information on evenings, central traffic). The corresponding data requirements are general city information, restaurants, traffic, culture, business clubs, good dining addresses, and the booking service. The story can start with a brief survey on the city and places to see. Next we consider the selection of a hotel, surveys on events of interest, bookings for events, search for dining, what else to see, whom to talk to. The specific utilization is based on basket, shallow navigation, fast search, and highlight hopping.

Before developing the storyboard let us refine the user side in [Figure 11.6](#). [Figure 13.6](#) is an extension that can be used for a more detailed specification of the presentation system under consideration of the social aspects:

Profile: The user has a specific *profile* with characteristics of user with their knowledge and abilities, etc.

Context includes also the *society* in which a user is a member.

Portfolio associates the tasks a user has to perform with the life cases under which a task becomes essential.

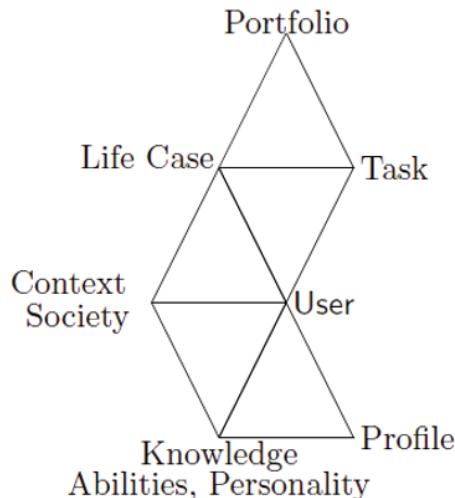


Fig. 13.6. Application systems are user-oriented: The user viewpoint with profile and portfolio

We may now separate the presentation system specification into
prescription of the presentation system on basis of the storyboard with a corresponding refinement of the storyboard and
specification of the layout system on the basis of principles of screenography and characteristics of users.

Additionally the layout system can be specified.

Prescription of the Storyboard

Step 5. Storyboard prescription	<ol style="list-style-type: none"> 1. Story space elicitation Detailed modelling of tasks 2. Description of actions Refinement of tasks Refinement of functions Specification of processes Dynamic integrity constraints 3. Refinement of storyboard prescriptions Quality requirements and strategies for their satisfaction Reconsideration and resolution of obligations Completeness check for media types and activities Specification of activity phases Derivation of requirements to media types
Precondition	Agreed portfolio with application departments Corporate identity known Business processes completely known Dependence of business processes completely known Organization models saturated Accepted quality model
Postcondition	Agreed task portfolio Restrictions for usability Obligations for functionality at business user layer satisfied Quality manual Obligations for development and implementation Obligations for interactivity of business user layer

Scenario specification and integration is based on dialogue scenes which consist of steps with a plot. Dialogue scenes represent an episode. Actor are supported with their information need and interaction requirements. Interaction objects satisfy these interaction requirements and information need. These stories may now be cut to *mini-stories*. A *mini-story* typically captures a small, self-contained, tightly connected set of scenes similar to a movie clip.

Stories and scenarios are parts of the story space. Composition of scenarios and stories within the story space may use methods developed for case-based reasoning and decision theory in the form

action if condition, e.g.,

CultureEntry if CultureSelect; EventsEntry if EventsSelect.

For instance, a scenario for a community website might be the *membership application*.

Membership Application; Purchase with Approval

Intention: Actor wants to apply for membership

Roles: Actor as applicant

Systems: Electronic Commerce Web Server, Certification Authority, ...

Scenario

Dialogue Step: Making Membership Application Available

Goal: Membership Application Available

Outcome: Membership Application is available

Actions: Actor invokes the membership application page

....

Dialogue Scene: Approving Payment

Occasion: Membership requires payment first
 Goal: Association budget balance increased
 Outcome: Certification authority has approved
 membership fee payment
 Information need: Association account, bank details
 Actions: Membership data are processed
 Web server receives payment information
 Certification authority consulted
 Certification authority approves payment
 Web server notifies actor
 Context: safe connection, existing application
 ...

The prescription of the storyboard integrates the description of the intention (main goals, minor intentions), scenarios as expressions on dialogue actions, actors within their role of the actor according to the task, communication among actors, content and functions, context, and tasks to be performed in the current step.

For instance, an infotainment website for a city may include intentions such as attracting visitors and providing information for inhabitants. Main actors are tourists, inhabitants, and business people. Travel information scenarios are based on separation depending on the kind of user, context and preliminary steps. A typical scenario is a travel from A to B with search for closest stopping places and routes, with input of dates (in general, day, daytime, ...) and input of travel preferences (time, attractiveness, convenience) followed by a selection of preferred choices. A schedule export with different choices might contain detailed travel plan with(out) sightseeing points, side conditions, transfer description, with graphical plans and export file options. It might be concluded with booking and reservation offering, return travel, etc. Another scenario is a round-trip travel from point A or reaching trains.

Actors within the travel information scenario are, for instance, naive users that are unexperienced with area, city, do not know specifics, have positive and negative experience w.r.t. environment, language. Therefore, robustness, expressibility, and sophisticated representation techniques are essential. These actors act with restricted attention according to their polarity profile (psychographical profile). They are interested in objective and relevant information and not in narrative information. Their profile might be classical (and not hyper-modern), between tough and tender, more conventional than funny, more traditional than avant-garde, more natural and simple than artificial and complex, introverted or extroverted, more stable than labile, etc. Their physical profile ranges from sportsman to handicapped.

In a similar form we can specify visiting a town for the first time (information need profile: simple and easy travel, survey, places of main interest, profile of the town, relaxation; specific services applicable: highlights, hot spots, very recent news, special offers; typical demographical profile).

The development of an infotainment site brought up around 30 similar scenarios for city information sites similar to the naive actor or to the Jack-of-all-trades scenario that we used in previous chapters.

Based on the scenarios and their integration we can develop a high-level blueprint. The information architecture of the site uses a rough navigation structure (organization of the site), labeling of the major areas (beginning with a bird's-eye

view) and sub-sites as with specific characteristics (title (for fast retrieval, bookmarking), description (for text search), keywords (for ontology or thesaurus search), audience (for actor adaptation), format (for download, etc., style, representation), topic (area, ontology items)). The blueprint focuses on major areas of the site and ignores specific navigation or page-level details. The decoupage of the site adds activities of actors in areas and dialogues of actors in areas.

The blueprint is enhanced by:

- Guides for introducing new actors to the site, to main content, to organization, to specific service;
- Search and browse features for fast access to the contents;
- Sub-site directories as catalog of the site;
- Sub-sites containing the actual content of the site;
- Grouping of similar pages (e.g., in sub-sites) or related pages (e.g., with similar treatment);
- Billboards for dynamic news, specific highlights, news and announcements with links.

The decoupage can, for instance, be represented in a table form as follows: for search and retrieval for events:

page	actor	activity	dialogue	interaction object
p1	a1, a2	selection of kind	selection dialogue	m1
p1	a1	choice of search style	tick on one of the styles	m1
p1	a1	extended search	selection of search areas composition of search expression	m1 m1

We trace utilization scenarios and derive then mockups using the utilization scenarios for check of the decoupage. Traces are assessed according to specific requirements, quality parameters, walk-trough test and run-through tests. Mockups are essentially quick and dirty textual documents (content, functionality, links of interaction objects) for discussion of implications of the decoupage. We may see the site in action before any code is implemented and apply basic usability tests. These are necessary (at least) for the major pages. We can check possible integration of organization, labeling, navigation, (visual) grouping of sub-objects, and rough styling.

Furthermore, we evaluate the decoupage, blueprint and scenarios by:

- Comparison with the competitors (features of similar pages, plans for development; success and failure of competitors; good ideas, good content). We may refine the project plan with its development steps and time restrictions. We assign tasks to team members and consider outsourcing. Project costs can now be reestimated (investment costs (development, hardware, software, marketing, imported content); running costs (provider, hardware, software, support, marketing, feedback, imported content)). So, the project may also be downsized by dividing the project tasks into categories (must, optional, nice to have), re-developing the project plan and re-calculating the costs, and plan for stepwise introduction or delivery.

The co-design aspects and the partner dimension do not change much. We can use the same forms as in the previous subsection. Methods and heuristics vary however.

Methods and heuristics	Observation of the reality
	Case studies for similar applications
	Interview techniques
	Usability studies
	Rephrasing of application conceptions
	Detection techniques for exceptions
	Tracing utilization scenarios
	Evaluation of the decoupage

The literature on usability is fairly rich. Heuristics may be combined with guiding principles like the following ones which have already been used in the early 1990s:

- Clear description on potential users and their tasks;
- Help and guide the user during derivation of a mental model;
- Use the language of the user;
- Display systems states in a distinguishable form;
- Clear specification of possible actions;
- Clear structuring of the user interface;
- Realization of distinguishable feedback;
- Adaptation of interfaces to user profiles, cognitive skills, environment;
- Combination of visual interaction with language-based interaction;
- Restrict requirements to users' memory;
- Allow to disable and undo actions and previous steps;
- Support in recognition and diagnosis of errors, failures and wrong steps;
- Do not surprise the user;
- Main interaction steps are supported in best practices approaches;
- Explain usage of site by examples and formalism.

Specification languages	HERM, SiteLang, DistrLang Description of human-machine behavior
Infrastructure supporting the step	Storyboard editor, e.g., ADOxx Word processor Generators (XSL ...)
Used additional documents	Regulations/laws/restrictions Style guides Presentation pattern, grid libraries

Specification of the Layout System

Step 6. Presentation system layout specification	1. Develop a general screenography style specific style guide requirement to graphical user interfaces principles of visual representation
	2. Derive rules for the entry page for visual design grid geometry
	3. Revise schedule and budget constraints
	4. Map to collage techniques
	5. Refine cognitive aspects
	6. Combine all elements into the WIS screenography

The step is based on the considerations for screenography. The screenography style incorporates

- graphical style guides for graphical user interfaces which are aesthetically pleasing, clear, compatible, comprehensible, efficient, consistent, controlled, direct and intuitive, flexible, forgetful of failures, predictable, recovery-supporting, responsive, simple, transparent, familiar, and do not have trade-offs;
- specific corporate-oriented style guide for interaction object, the web page portfolio, for graphical items, for functionality and especially navigation and search, and for application generation;
- criteria for visual representation such as elegance/simplicity, scale, contrast and proportion, organisation and visual structure, grid-based design, imaging and representation, and styling characteristics of means, values, and cultures;
- typography guidelines for readability, simplicity, proportions, sizing, text styles, contrasting, texture and dynamics, representation of lines and points and figures, and means for representation;
- colouring guidelines according to the website identity and based on the science and art of colouring;
- guides for contrasting and rhythm of the pages and the site.

The *entry page* is the main entrance to the website. For this reason, its development must be especially carefully developed. It is similar to the exposition sequence in movies. It conditions and orients the visitor. At the same time, the entry page should also conform with the style guideline of the entire website. The entry page is based on communication psychology and rhetoric, i.e., it should answer the seven questions *who, what, when, where, why, in what way, by what means* in a convincing graphical design. Visitors believe that he/she finds what he needs. Suppliers provide their intention in a way that convinces about their trustworthiness and reliability. Conclusions one might draw from the entry page are coherent with the entire website. The impression is that the site is professional and of high quality. In general, the entry page can be structured according to the communication act (see [Figure 5.2](#)).

The entry page follows the *oneness paradigm*, i.e., presentation, topics, effects, aesthetics or poetics, and the technical medium form a holistic unit. The entry page displays a variety of messages about the content according to the information need, about the functionality that supports activities of the user, and the story space users might face. The visitor should be able to draw the right conclusions already while interpreting the entry page. It is a good rule to be spartan and to use a minimal set of elements. All fancy and moving elements should be removed.

The entry page is also a self-manifestation of the supplier or provider with respect to the relationship to the users. Entry pages are often overdone and display whatever a developer can use for web page programming. The specific style of overloading is one of the main errors that web developers can make.

Identity sites as well as most other websites combine the five types of identity. The actual identity shows what the organisation/institution/personality is. The communicated identity is based on the way how the organisation/institution/-personality communicates its identity, internally and externally. The conceived identity is generated by the user and shows how the organisation/institution/personality is perceived externally and internally as a result of both its actions and communications. The ideal identity targets the optimum positioning of the organisation/-institution/personality in its market or markets taking cognisance of its strengths

and abilities in addition to environmental considerations. The desired identity is the identity which the chief executive and management board wishes to acquire.

Collage techniques are graphical techniques for combination of content. They may also be extended by integration of stories, functionality and context. They have a high expressive power and support a holistic definition of identity of a website. These techniques relate to reality by providing the right display, relating to the most similar metaphor, delivering by the best colouring and texture schema, and using an optimal organisation of the combination of its elements to their users. Collages have a theme, a master profile and an idea. Their elements are combined in such a way that the presentation becomes holistic. These techniques follow the logics of part-whole and the principles of fortuitousness and synchronism.

13.5 WIS Specification (Design and Development)

The design and development phase of a WIS development is now mainly targeting on the middle part in [Figure 13.6](#): the service interface. We must however also develop the technical system. For development of the latter one we can however rely on the literature that discusses in detail (i) database design and development, (ii) workflow design and development, (iii) technical system development and programming, and (iv) technical infrastructure development and programming. We do not consider in this monograph aspects of maintenance, migration and evolution, testing, deployment, delivery, etc.

As we mainly discussed the presentation system we do not repeat the large body of knowledge for database development, content management, workflow management, etc. Instead we sketch the transformation and development steps and refer to development guides. The order of activities has however to be changed. Classical information system development starts with the development of the database schema and of views on top of the database schema. The database algebra is taken for granted. The database schema and views are going to be mapped to the logical schemata and view collections. After enhancement these schemata and views are mapped to physical schemata, support features, physical views, etc. Views are also *consumption views* and *production views* for BPMN activities or more general for BPMN flow object. Now we can develop workflow diagrams and map these diagrams to realisations. Realisation can be programs of a workflow engine or programs in a programming language. A program is then supporting each action a user might perform. Programs might have their call interface for user involvement. We advocate the QuASAR framework. Therefore the corresponding application architecture represents the features that are provided for the presentation system. The presentation system has its interfaces for user interaction.

This approach is not entirely appropriate for web information systems. We might revise this approach and use the following phases:

1. *Storyboard specification*
2. *Development of workflow suites*
3. *Development of database and view schemata*
4. *Development of database schemata*
5. *Transformation of the conceptual co-design schema to logical schemata*
6. *Program development*

7. *Development of supporting means*
8. *Development of a database farm as integrated solution*
9. *Integration of security and privacy*

The last five phases are out of the scope of this book.

Goals and subject	Specification of the conceptual HERM schema including integrity constraints Specification of media types and containers Detailed development of data types Validation of structuring Utilization pragmatics of all data types Attribute type hierarchy Detection of all real life constraints Specification of harmonized phase-based enforcement schemes Decomposition into an easy to maintain schema Functionality and workflows completed Understandability of application behavior Identification frames and obligations State transition diagrams of all actions and activities Mappings of associations between conceptual and external schemata Collaboration schemes
Developed documents	Conceptual schema and external schemata View collaboration Enforcement schemes for integrity constraints Basic data types with attribute assignment Derivation schemes Workflow engine workflow and processes Performance parameter of functions Media types Containers Application architecture (with integration into TI- and T-architecture)
Revised documents	Conceptual schema including integrity constraints View schemata Storyboard Specification of media types Enforcement schemes Workflow specification Specification of media types Translation profile for operations Design obligations Schema translation profiles

The production of the web information system may be based on the following issues:

- Deriving implications of

- actors profiles for presentation of content, for functionality (incl. help), for context;
- platform and software selection for programming and environment (restrictions);
- content and functionality requirements for media objects.
- Story specification for navigation, search and retrieval, export/import of data, for interaction support.
- Development of detailed blueprints.
- Content mapping.
- Web page inventory and management of production.
- Developing site style guides.
- Quality assurance, documentation.

This approach allows us to derive *production rules*. The production process is supported by developing rules for team development, development and implementation steps, production, time tracking, asset management, automatic generation, team task planning, and content acquisition. The customer communication should be based on regular meetings, memos and reports, resets and meltdowns similar to agile methodologies. Test methods such as program and database tests or site tests should be continuously applied. The site can be understood as a database with version control and libraries, with management of interaction objects, and with data acquisition planning. Installation and maintenance plans include issues such as staging of the site, provider planning, mirror, documentation, juristic problems, load test planning, hosting, maintenance, licensing, contracting, content management, and re-development planning.

It is a good style to clarify conventions such as naming conventions (for programs, source code, database structures), metadata integration (developer identity, content hint, version), comment conventions (internal, external and in-use comments for comment extractors (e.g., JavaDoc)), and website structuring conventions (e.g., search and navigation conventions for holistic navigation and extension; also input/output conventions).

Storyboard Development

Step 7. Storyboard specification	<ol style="list-style-type: none"> 1. Detailed specification of the story space <ul style="list-style-type: none"> Development of mini-stories Detailed development of the discourse Description of collaboration frames 2. Application of style guides <ul style="list-style-type: none"> Specification of presentation grids 3. Derivation of the storyboard <ul style="list-style-type: none"> Specification of dialogue scenes Prescription of media types Prescription of containers Development of associations <ul style="list-style-type: none"> among dialogue steps and scenes Categorization of main media types and subtypes Specification of dialogue scenes 4. Development and validation of mockups <ul style="list-style-type: none"> Development of a high-level blueprint Sketching mockups Tracing utilisation scenarios Design sketches and storyboard development
---	--

Precondition	Finalized style guides Agreed dialogue environment Architectures completed Utilization restrictions, barriers of utilization
Postcondition	Completed storyboard Completed scenes Generic Profile and completed portfolio of all actors Mappings of all business processes to plots Obligations for media types Rough specification dynamic IC Actions/activity sequences Obligations for development and implementation Commonly agreed mockups

The storyboard specification is guided by Chapters 1-11. We derive the entire story space. We shall later develop a macro-structuring for this story space. It is a good idea to develop the storyboard in the same manner as in agile programming, i.e., within a tight collaboration with business users.

The storyboard integrates structuring, linking and navigation, functionality such as search, and guidelines for the screenography. The structuring schema has to harmonise organization schemata for simplification of search engine (known item search) (exact organization schemes (alphabetical, chronological, geographical); ambiguous organization schemes based on classification schemes (topical, task-oriented, audience-specific, metaphor-driven); hybrid organization schemata). The structuring display is based on abstraction hierarchies, categories, display decisions for the basic media type (audio, etc.). Options for structuring are dependent on granularity, on relationships among categories (identical, extended, overlapping, restricting, without), and on level of detail (same, refining, abstracting).

The content structure follows hierarchical, hypertext or database structuring. Content partitioning depends on text length, content authoring, and web page adhesion. We sketch also quality assurance and presentation styles. In a similar way we can integrate stereotypes and pattern for navigation and linking. Linking must be supported by explicit maintenance. We distinguish between intra-interaction-object, viewpoint, traversal, meta, foreign-content, inter-interaction-object, context and actor links. They require different support features. It is a good trick to generate labels as shortcut label, reference label and identity labels.

Usable input documents	WIS development and system documentation Application conceptions, forms, etc. Architectures of the application Story space prescription and WIS utilisation Treatments of the story space Activity documentation and frames Actor portfolio and actor profiles Obligations for media types Conceptual schemata Task manuals Rough presentation documentation and frames Presentation templates and pattern Rough collaboration documentation Collaboration frames, presentation frames
------------------------	---

Theoretical fundamentals	Theory of media types Collaboration sciences, theory and psychology Psychology Dramaturgy, scenography
Methods and heuristics	Observation of the reality Case studies for similar applications Interview techniques Usability studies Rephrasing of application conceptions Detection techniques for exceptions Tracing utilization scenarios Evaluation of the decoupage
Specification languages	HERM, SiteLang, DistrLang Description of human-machine behavior
Infrastructure supporting the step	Storyboard editor e.g., ADOxx Word processor Generators (XSL ...)
Used additional documents	Regulations/laws/restrictions Style guides Presentation pattern, grid libraries

Development of a Workflow Suite

The story space follows the global-as-design paradigm. A BPMN workflow specification follows the local-as-design paradigm. We cannot use the database approach by developing a global schema and by representing local viewpoints as views on the global schema. Therefore we have to use a local representation for all traces. We could use a full generation of all potential stories. Instead, we concentrate development on potential scenarios. We use the mini-story clustering as a zoom mechanism. A storyboard is going to be represented as a macro-storyboard. The scenarios that have been developed can be directly mapped to macro-scenarios. Each macro-scenario can now be represented as a set of BPMN workflows and mainly BPMN process diagrams.

A collection of BPMN diagrams (process, collaboration, communication and conversation) becomes a *BPMN diagram suite* after enhancement by explicit specification of the following elements

- association or collaboration schema among the diagrams,
- controllers that maintain consistency or coherence of the model suite,
- application schemata for explicit maintenance and evolution of the model suite, and
- tracers for the establishment of the coherence.

Step 8. BPMN transformation of scenarios	<ol style="list-style-type: none"> 1. Wrapping mini-stories with start and end scenes 2. Development of macro-storyboards with mini-stories as chunks 3. Elaboration of scenarios within the macro-storyboard as macro-scenarios 4. Transformation of macro-scenarios into BPMN workflows with complex activities 5. Transformation of mini-stories into suites of BPMN diagrams 6. Development of collaboration diagrams on the basis of the macro-storyboard and the macro-scenarios 7. Development of conversation diagrams on the basis of the macro-storyboard and the macro-scenarios 8. Development of conversation diagrams on the basis of the macro-storyboard and the macro-scenario 9. Derivation of views for each data-based BPMN flow object
---	--

Development of Database and View Schemata

So far, we have got a list of things of interest for our database support. Database management systems follow a global-as-design paradigm. Therefore, we develop now first a database schema and based on that view schemata.

Precondition	Complete knowledge on all application data Glossary of application notations
Postcondition	Complete type structure of views, Mappings between schemata and views Obligations for structuring at business user layer satisfied

The database development may follow the classical approach to co-design of database systems. Therefore, we shall use now the following steps for database and view development:

Step 9. Description of kernel types	<ol style="list-style-type: none"> 1. Development of kernel types based on application conceptions 2. Grouping of types 3. Development of type hierarchies 4. Description of behavior for types 5. Development of cluster types 6. Detection of objectification borderline
--	--

Step 10.	1. Identification for kernel types 2. Identification within hierarchies 3. Identification candidates in relationship types 4. Association of keys to component types 5. Recursive definition of the identification 6. Identification for cluster types
Development of identification frames for all ER types	
Step 11.	1. Completion of attribute structure possibly with interfaces to other applications 2. Completion of semantics 3. HERM representation of types 4. Refinement of types 5. Check of type 6. Check of hierarchies 7. Modular refinement of units
Finalizing type specification	
Step 12.	1. Specification of existence constraints 2. Enforcement schemes for insert operations for delete operations for update operations 3. Consistency check
Handling mechanisms for identification of types	
Step 13.	1. Development of sample populations 2. Modeling of semantics 3. Check of structure and static semantics 4. Decomposition and refinement of types 5. Check through sample refinement
Validation of view schemata based on sample data	
Step 14.	1. Validation of attributes structure 2. Revision through hierarchical decomposition 3. Revision through semantical decomposition based on FD's 4. Treatment of key-based and other IC's 5. Transparent decomposition for tuning 6. Control of problematic units 7. Revision through semantic decomposition based on JD's 8. Revision of the operational behavior
Validation of views through normalization	
Step 15.	1. Collection of domain types and their characteristics 2. Semantics of domain types 3. Domain types for keys and integrity constraints 4. Rules for derived attributes and concepts 5. Domain types in hierarchies 6. Consistency rules for consistency maintenance
Specification of domain types	

Step 16. Finalizing functionality specification	<ol style="list-style-type: none"> 1. Description of the workflow engine and separation of functionality 2. Specification of database transitions and special treatment of transactions 3. Specification of query functions and modification operations 4. Specification of dynamic integrity constraints and development of maintenance routines 5. Support for media types 6. Development of workflow fields and generalization to generic functions
Step 17. Integration of user views	<ol style="list-style-type: none"> 1. Inter-semantics of views 2. Incremental integration of view schemata <ul style="list-style-type: none"> a. notation associations of all schema concepts b. attribute, entity and relationship types c. hierarchies d. translation of functionality, partial functions e. translation of semantics, partial IC f. derivation of schema-view mappings 3. Normalization 4. Collaboration associations: Mappings 5. Stability analysis and extension 6. Integration of dialogues and views

Steps 10-18 are based on co-design for classical information systems. Therefore, we omitted the detailed consideration.

Transformation of the Conceptual Schemata

We sketch now the last step within the scope of this monograph. This step relies on the co-design approach. We start with the transformation of conceptual database schema to the corresponding logical schema. It is a good trick to enhance the schema by directives that provide hints for the translation. We can use a stereotype-pattern-template approach for the transformation.

Step 18. Transformation of the conceptual schemata of	<ol style="list-style-type: none"> 1. Translation (directive-based) of HERM schema and functions Derivation of the view suite Settling value types and value maintenance Settlement of integrity control Workflow mapping Protocol and stub templating Script template derivation 2. Screenography template population Titivation of pages, functionality 3. Interaction type schema derivation Layered and folded schemata Container design Metaphor integration Security and privacy towers 4. Performance profiling and adaption Throughput analysis Platform directives Controlled and robust redundancy Tuning Evolution forecasting and preparation 5. Usability analysis Assessment Diagnostics Systematic empiric testing
--	--

Quality of Service Assurance for WIS

Quality of WIS is characterized depending on the abstraction layers:

Quality parameters at business user layer may include *ubiquity* (access unrestricted in time and space) and *security/privacy* (against failures, attacks, errors; trustworthy; privacy maintenance).

Quality parameters at conceptual layer subsume *interpretability* (formal framework for interpretation) and *consistency* (of data and functions).

Quality parameters at implementation layer include *durability* (access to the entire information unless it is explicitly overwritten), *robustness* (based on a failure model for resilience, conflicts, and persistency), *performance* (depending on the cost model, response time and throughput), and *scalability* (to changes in services, number of clients and servers).

We use a number of measures that define quality of service (QoS) for WIS:

- *Deadline Miss Ratio of User Transactions:* In a WIS QoS specification, a developer can specify the target deadline miss ratio that can be tolerated for a specific real-time application.
- *Data Freshness:* We categorize data freshness into database freshness and perceived freshness. Database freshness is the ratio of fresh data to the entire temporal data in a database. Perceived freshness is the ratio of fresh data accessed to

the total data accessed by timely transactions - transactions which finish within their deadlines.

- *Overshoot* is the worst-case system performance in the transient system state. In this paper, it is considered the highest miss ratio over the miss ratio threshold in the transient state. In general, a high transient miss ratio may imply a loss of profit in e-commerce.
- *Settling time* is the time for the transient overshoot to decay and reach the steady state performance.
- *Freshness of Derived Data*: To maintain the freshness, a derived data object has to be recomputed as the related ground database changes. A re-computation of derived data can be relatively expensive compared to a base data update.
- *Differentiated Timeliness*: In WIS QoS requirements, relative response time between service classes can be specified. For example, relative response time can be specified as 1:2 between premium and basic classes.

We observe that these quality of services characteristics are difficult to specify in systems if architecture is not taken into consideration. Let us consider data freshness as an example for WIS. Data freshness results are related to *information logistics* that aims in providing the correct data at the best point of time, in the agreed format and quality for the right user at the right location and context. Methods for achieving the logistics goals are the analysis of the information demand, storyboarding of the WIS, an intelligent information system, the optimization of the flow of data and the technical and organizational flexibility. Therefore, data freshness can be considered to be a measure for appropriateness of the system. Depending on the requested data freshness we derive the right architecture of the system.

Based on our co-design modelling approach and as a result of separation of concern within the software engineering quadruple we can derive a number of techniques for architecture-driven and application-domain-rules modelling of high quality WIS:

- *Introduction of artificial bottlenecks*: Instead of replicating data at different sites or databases we may introduce a central data store that exhibits a number of versions to each of the clients that require different data.
- *Introduction of a tolerance model*: We may introduce an explicit tolerance model that decreases the burden of data actuality to those web pages for which complete actuality is essential.
- *A cost-benefit model of updates*: Updates may sometimes cause a large overhead of internal computing due to constraint maintenance and due to propagation of the update to all derived data. We thus may introduce delays of updates and specific update obligations for certain time points. Typical resulting techniques are *dynamic adaptation of updates* and the explicit treatment by an *update policy*.
- *Data replication in a distributed environment*: Data access can be limited in networking environments. The architecture may however introduce explicit data replication and specific update models for websites.

This list of techniques is not complete but demonstrates the potential of architecture-driven WIS development.

13.6 Bibliographical Remarks

13.6.1 WIS Development

Web information system engineering has attracted a lot of research. There are many techniques that go beyond software engineering. The system specification approaches developed by [97, 98, 314] are the software engineering basis for our approaches. Other approaches to web engineering are HDM [269, 761], IDEA [143], IFML [340], Integranovo [343], MagicUWE [883, 884], M2M [572], MOF [572, 629], MOF2T [572], OOHDM [686, 684, 762, 763], PIM (platform independent modelling, e.g., [589, 688]), PSM (platform specific modelling, e.g., [112, 276]), RMM [345], Rux-Tools [658], UWA [101], UWE [102, 884], WAE [157, 485], WebML [906, 116, 144], WebRatio [907, 116], WSDL [182, 183, 766], and WebSpec [521]. The storyboard approach is not covered by these methodologies. Most of them are influenced by methods of object-oriented and UML design techniques. The co-design of web applications is mainly oriented on content modelling and modelling of some (main) functional features such as navigation.

Model-driven development, generative programming, and model-driven software engineering are alternatives to our technique, e.g., [170, 553, 587, 660, 928]. Websites are constantly evolving, must maintain their quality, and conform to security and privacy concerns. We left these topics outside of this monograph and refer to specific books and papers, e.g., [13, 358, 478, 647, 746, 943].

Let us concentrate on some of the main approaches:

ARIS (*Architecture of Integrated Information Systems*) [700] defines a framework with five views (functional, organizational, data, product, controlling) and three layers (conceptual ('Fachkonzept'), technical ('DV-Konzept'), and implementation). ARIS was designed as a general architecture for information systems in enterprise environments. Therefore, it is too general to cover directly the specifics of Web information systems and needs to be tailored.

The *Rational Unified Process* (RUP) [465] is an iterative methodology incorporating different interleaving development phases. RUP is backed by sets of development tools. RUP is strongly bound to the Unified Modelling Language (UML). Therefore, RUP limits the capabilities of customization. Like ARIS, RUP does not address the specifics of WIS-E. A similar discussion can be made for other general purpose approaches from software engineering.

OOHDM [763] is a methodology which deals with WIS-E specifics. It defines an iterative process with five subsequent activities: requirements gathering, conceptual design, navigational design, abstract interface design, and implementation. OOHDM considers Web Applications to be hypermedia applications. Therefore, it assumes an inherent navigational structure which is derived from the conceptual model of the application domain. This is a valid assumption for data-driven (hypermedia-driven) Web applications but does not fit the requirements for Web information systems with dominating interactive components (e.g., entertainment sites) or process-driven applications. There are several other methodologies similar to OOHDM. Like OOHDM, most of these methodologies agree in an iterative process with a strict top-down ordering of steps in each phase. Surprisingly, most of these methodologies consider the implementation step as an 'obvious' one which is done by the way, although specifics of Web applications cause several pitfalls for the unexperienced programmer espe-

cially in the implementation step. Knowledge management during the development cycles is usually neglected.

There are several methodologies that cope with personalization of WISs. For example, the HERA methodology [333] provides a model-driven specification framework for personalized WIS supporting automated generation of presentation for different channels, integration and transformation of distributed data and integration of Semantic Web technologies. Although some methodologies provide a solid ground for WIS-E, there is still a need for enhancing the possibilities for specifying the interaction space of the Web information system, especially interaction stories based on the portfolio of personal tasks and goals.

This list of projects is not complete. Most of the projects are not supporting conceptual development but provide services for presentation layout or playout. The *Yahoo pipes* project¹ uses mashup services for remixing popular feed types. The *Active Record* pattern embeds the knowledge of how to interact with the database directly into the class performing the interaction.

There are many guides to software and systems architecture, e.g., [774, 57, 60, 103, 780]. The Quality Software Architecture (QuaSAR) framework has been introduced by [783]. We use [359, 415, 414, 681] for our approach.

The service specification framework has been introduced in [175]. The rhetorical frame is an old concept. It dates back to Cicero and even to Hermagoras of Temnos who was one of the inventors of rhetoric frames in the second century BC. The latter used a frame consisting of the seven questions: Quis, quid, quando, ubi, cur, quem ad modum, quibus adminiculis (W7: Who, what, when, where, why, in what way, by what means). The work of Hermagoras of Temnos is almost lost. He had a great influence on orality due to his proposals, e.g., [262, 814, 922]. For instance, Cicero [922] has intensively discussed his proposals and made them thus available. A later reinvention is Zachman's framework as an inquiry system for information systems engineering [900, 899, 939] and the w-framework by Kipling.

Hook and anchor techniques have been revisited for metadata development in [457, 458, 459]. Memes have been elaborated for IT systems in [815]. Classically memes have been considered as structures that are encoded within a gene or a suite of genes. We used a notion of information that is better fitted to the needs of information systems and of content systems. It bases the existence of information for a user on this user's abilities for perception (1), abilities for selection (2), the interests (3), the knowledge obtained so far (4), and abilities for integration (5). This notion nicely corresponds to different uses of information as noted in [815]: generation, externalization, recording, protection, communication, distribution, sharing, referencing, editing, search, analysis, management, and annihilation.

Knowledge chunks [365, 409] generalise the content-concept-topic approach [104, 115, 223, 227, 467, 513, 514, 947, 593, 724, 832, 837, 837]. The architecture framework we used has been discussed in detail in [359]. Requirement engineering is an area of intensive research [337, 382, 654, 918]

WIS engineering generalises classical software engineering, e.g., [25, 11, 246, 267, 275, 337, 382, 654, 672]. The SPICE framework [224, 349, 346, 499] can be applied to web information system engineering. Requirements engineering is a well-established and intensive field of research (see for instance, [97, 337, 366, 384, 654, 917, 934]). The approach we presented follows those classical approaches.

¹ See: <http://pipes.yahoo.com>

The 3U framework relates back to [628, 808]. The SMART framework has been introduced in [851, 841, 846]. It enhances approaches to sophisticated service systems, e.g., [230, 720]. The given heuristics for usability are still important. One might compare the top ten mistakes of web design given in 1999 and given in 2016. It is shocking that the lists almost coincide.

A co-design workbench is available within the ADOxx metamodeling platform [6, 460]. It includes development of HERM schemata, of well-formed BPMN diagram suites and of storyboards. It also provides translation services. This implementation can easily be extended on the basis of the AdoScript language and the AQL query language.

We discussed three wiki functionalities that are backed by our approach. Presentation may also include generic adaptation to the user environment and features for marking content [585]. Wiki systems should integrate features that have been developed for customer management. Wikis are generally designed with a functionality that makes it easy to correct mistakes. Since this functionality is a target for attacks on content and on the system, wiki systems are extended by security and quality management features. Thus, they provide a means to verify the validity of recent additions, changes, corrections, replacements, etc. to the content. History and development information can be maintained through dockets [836] and the diff feature that highlights changes between two revisions. Wiki systems are special web information systems. They support information seeking life cases [204, 732], are based on storyboards for creation and consumption of information [727] and require a sophisticated user profile and portfolio model [735].

The collaboration story space for the paper submission and reviewing scene MuConMS (multiple conferences management system) is based on [418].

Stereotypes can be understood as general solutions to a problem. Pattern and templates are refinements. This refinement approach is known under different notations: inverse modelling [564] with parameter instantiation, calibration and optimisation of the model; generic models represent a parameter-based class of more specific models [86, 872]; represent a class of workflows with the same flow of action in the same application area. Model-driven architectures and universal applications [803] use some kind of top-down technique for refinement of models; universal applications [617] use generators for derivation of the specific application; concept spaces [17] use generic elements or concepts for construction; data mining and analysis starts with suppositions which model class might reflect the current situation; pattern-based reasoning [17] uses generalised solutions for a class of problems and adaptation or refinement methods; inductive learning [942] starts with a supposition that a certain kind of explanation can be used; patient modelling [462] starts with a very general model of combinations of sicknesses and adapts the patient-specific, individualised model in accordance to the observations and data.

The stereotype-pattern-template framework has been elaborated in [17]. Presentation design may inherit knowledge gained in art, staging and stage setting, architecture (e.g., [775] or approaches by Vitruvius and L. B. Alberti), interior design, industrial design, and human user interface design (e.g., [592]). Entry page design should be performed by graphics specialists. The literature is fairly rich, e.g., [474, 582]. Mini-stories are considered in [244, 871].

Model suites [173, 756, 793, 794, 839] have been used as the basis for BPMN diagram suites. Quality assurance is an important problem for WIS as well. Our view is based on [356, 357, 358].

The extended entity-relationship model has been deeply investigated in [823]. The corresponding methodology is presented there as well.

13.6.2 Realisation of Web Information Systems

The introduced architecture for WIS is based on advanced database application architectures, e.g., [42, 303, 414, 442, 512]. Many aspects discussed in this monograph have been realised in infotainment WIS such as www.cottbus.de [757, 765]² and institutional websites, in e-government WIS such as SeSAM [226], in edutainment system such as DaMiT (Data Mining Tutor) [92, 371] or in community WIS such as conference management systems [418, 719] or websites of associations, e.g., the Cottbusnet association www.cottbusnet.de, including electronic set-top-box cable net portal CBI including electronic personalised program guides (with facilities for video-on-demand, TV/radio-on-demand, EPG-on-profile, internet-on-profile, and phone-on-portfolio), in disaster management systems, e.g., [871, 872] for INDYCO, and in e-business WIS as inner portals for procurement, planning, distribution, accounting, and material management. The material in this monograph has been used for WIS courses in Austria, Chile, Finland, Germany, Kuwait, Japan, New Zealand, Russia, and United States.

There are hundreds of books on web programming, e.g., [117, 194, 232, 233, 247, 289, 616, 677, 764, 770, 776, 902, 920]. The three-score realisation projects in which we have been participating have had a preference to Linux as the operating system, to Perl as a programming language, to Sybase or later Oracle as the database management system, and SQL as database programming language. Many websites also use PHP, MySQL and various other operating systems. There exist also dozens of books on screen design tools starting with [598, 597]. A large number of technologies have been developed since the first websites, e.g., AJAX [22, 27, 58, 124, 216, 402, 648, 694]. Although Web 1.0 has already used technologies that later have been recognised as Web 2.0 or Web 3.0 techniques, website development must consider these new developments.

Key Messages

Systematic Development of Web Information Systems

- applies the co-design method for web information systems presented in Chapter 11;
- faces the co-design method with strategic aspects, storyboarding, web interaction types and screenography as discussed in all the preceding chapters;
- enhances the co-design method by concrete steps, which constitute a specific development process.

² We acknowledge the contribution of the Cottbus Information Services and Interactive Team, esp. S. Berg, C. Binder, G. Bogacz, W. Clauss, H.-J. Dosdall, T. Feyer, C. Galke, S. Hamann, B. Heinze, T. Kobienia, A. Krohn, J. Lewerenz, T. Mielke, G. Millahn, M. Radigk, T. Schmidt, S. Schoradt, R. Schwietzke, S. Srini-vasa, V. Vestenicky, J. Wölkerling, S. Yigitbasi, and to our company partners.

Bibliography

1. N. Abdullah, Y. Xu, and S. Geva. A Recommender System for Infrequent Purchased Products based on User Navigation and Product Review Data. In *WISE Workshops*, pages 13–26, 2010.
2. S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann Publishers, 2000.
3. S. Abiteboul and P. Kanellakis. Object identity as a query language primitive. In *Proceedings SIGMOD 1989*, 1989.
4. A. Achilleos, G. Kapitsaki, and G. A. Papadopoulos. A Model-Driven Framework for Developing Web Service Oriented Applications. In *ICWE Workshops*, volume 7059 of *Lecture Notes in Computer Science*, pages 181–195, 2011.
5. M. Adams. Dynamic workflow. In *Modern Business Process Automation*, pages 123–145. Springer, 2010.
6. ADOxx. The ADOxx metamodelling platform. <https://www.adoxx.org/live/home>, 2016. Accessed Dec. 10, 2016; see too <https://www.youtube.com/watch?v=W89bzUwxEPo>.
7. I. AG. Homepage. <http://www.infoasset.de>, 2016.
8. G. Agha. *Actors: A Model of Concurrent Computation in Distributed Systems*. MIT Press, Cambridge, MA, 1986.
9. R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional database. In *Proc. Data Engineering Conference, Birmingham*, pages 232–243, 1997.
10. R. S. Agrawal, J. Kiernan, and Y. Xu. Hippocratic databases. In *Proc. 28th VLDB*, pages 143–154, 2002.
11. J. A. Aguilar, A. Z. Colado, C. T. Barba, S. Misra, R. Bernal, and A. Ocegueda. An analysis of techniques and tools for requirements elicitation in model-driven web engineering methods. In *Proc. ICSSA 2015, Part IV*, volume 9158 of *Lecture Notes in Computer Science*, pages 518–527. Springer, 2015.
12. M. Akaishi, N. Spyros, and Y. Tanaka. A component-based application framework for context-driven information access. In H. Kangassalo, editor, *Information Modelling and Knowledge Bases*, volume XIII, pages 254–265. IOS Press, Amsterdam, 2002.
13. J. Akoka, L. Berti-Equille, O. Boucelma, M. Bouzeghoub, I. Comyn-Wattiau, M. Cosquer, V. Goasdouí-Thion, Z. Kedad, S. Nugier, V. Peralta, and S. Sisaid-Cherfi. A framework for quality evaluation in data integration systems. In *ICEIS (3)*, pages 170–175, 2007.

14. S. S. Al-Fedaghi. *Professional and Computer Ethics*. Kuwait University Press, 2003.
15. S. S. Al-Fedaghi. The “right to be let alone” and private information. In *7th Int. Conf. on Enterprise Information Systems, Miami*, 2005.
16. S. S. Al-Fedaghi. A systematic approach to anonymity. In *3rd Int. Workshop on Security in Information Systems WOSIS-2005, Miami, May*, 2005.
17. B. AlBdaiwi, R. Noack, and B. Thalheim. Pattern-based conceptual data modelling. In *Information Modelling and Knowledge Bases*, volume XXVI of *Frontiers in Artificial Intelligence and Applications*, 272, pages 1–20. IOS Press, 2014.
18. F. M. R. Alencar, J. Castro, G. A. C. Filho, and J. Mylopoulos. From early requirements modeled by the i* technique to later requirements modeled in precise UML. In *Proc. WER00*, pages 92–108, 2000.
19. C. Alexander. *A Pattern Language: Towns - Buildings - Construction*. Oxford University Press, 1977.
20. M. Ali and L. Brooks. Culture and IS: A criticism of predefined cultural archetypes studies. In *Proc. AMCIS 2008, Paper 62*. Association for Information Systems, 2008.
21. M. Ali and L. Brooks. Culture and IS: national cultural dimensions within the IS discipline. <https://core.ac.uk/download/pdf/335304.pdf?repositoryId=14>, 2008.
22. A. Alkaltun, P. Maisch, and B. Thalheim. *Interdisciplinary Advances in Adaptive and Intelligent Assistant Systems*, chapter Towards next generation web: Knowledge web, pages 1–25. Information Science Reference, Hershey, 2011.
23. M. Alkandari. *A Model of Multicultural Software Project Team Management applied in Requirements Engineering*. PhD thesis, Virginia Polytechnic Institute and State University, 2012.
24. J. Allwood. On the analysis of communicative action. *Gothenburg Papers in Theoretical Linguistics*, 38, 1978.
25. J. P. A. Almeida, M.-E. Jacob, and P. van Eck. Requirements traceability in model-driven development: Applying model and transformation conformance. *Information Systems Frontiers*, 9(4):327–342, 2007.
26. J. M. Almendros-Jimenez and L. Iribarne. User interaction and interface design with UML. In F. Ferri, editor, *Visual Languages for Interactive Computing: Definitions and Formalizations*, pages 328–356. Idea Group Reference, 2008.
27. G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services - Concepts, Architectures and Applications*. Springer, 2004.
28. A. Alturki, W. Bandara, and G. G. Gable. *Design Science Research and the Core of Information Systems*, pages 309–327. Lecture Notes in Computer Science 7286. Springer Berlin Heidelberg, Berlin, 2012.
29. M. Altus. User modelling for conceptual database design based on an extended entity relationship model: A preliminary study relationship model. In *Proc. ADBIS*, pages 46–51, 1996.
30. M. Altus. *Decision support for conceptual database design based on the evidence theory - An intelligent dialogue interface for conceptual database design*. PhD thesis, Faculty of Mathematics, Natural Sciences and Computer Science of BTU Cottbus, Cottbus, 2000.
31. M. Altus and B. Thalheim. Design by units and its graphical implementation. In U. Lipeck and R. Manthey, editors, *Proc. 4th GI-Workshop on Fundamentals of Databases*, Report ECRC-92-13, pages 31–49, 1992.

32. M. Altus and B. Thalheim. Strategieunterstützung in RADD. In F. Schweigert and E. Stickel, editors, *Informationstechnik und Organisation*, pages 137–152. Teubner-Verlag, 1995.
33. S. Amarakoon, A. Dahanayake, and B. Thalheim. A framework for modelling medical diagnosis and decision support services. *International Journal of Digital Information and Wireless Communications (IJDWC)*, 2(4):7–26, 2012.
34. S. W. Ambler. *Database Techniques: Effective Strategies for the Agile Software Developer*. John Wiley & Sons, 2003.
35. J. Anke and D. Sundaram. Personalization techniques and their application. In M. Khosrow-Pour, editor, *Encyclopedia of E-Commerce, E-Government, and Mobile Commerce*, pages 919–925. Idea Group Reference, 2006.
36. N. Aquino, J. Vanderdonckt, J. I. Panach, and O. Pastor. Conceptual modelling of interaction. In *Handbook of Conceptual Modeling - Theory, Practice, and Research Challenges*, pages 335–358. Springer, 2011.
37. P. Arató, Z. Á. Mann, and A. Orbán. *Component-Based Hardware-Software Co-design*, pages 169–183. Springer, Berlin, Heidelberg, 2004.
38. M. Arbib and M. Hesse. *The Construction of Reality*. Cambridge University Press, Cambridge, 1986.
39. C. Arellano, O. Díaz, and J. Iturrioz. Opening Personalization to Partners: An Architecture of Participation for Websites. In *ICWE*, volume 7387 of *Lecture Notes in Computer Science*, pages 91–105, 2012.
40. D. Arijon. *Die Grammatik der Filmsprache*. ZWEITAUSENDEINS, Frankfurt a.M., 2000.
41. Aristotle. *Poetics*. Hackett Publishing, Indianapolis, 1987.
42. H. Arndt. *Integrierte Informationsarchitektur - Die erfolgreiche Konzeption professioneller Websites*. X.media.press. Springer, 2006.
43. K.-H. Arnold. Didactics, didactic models and learning. In N. M. Seel, editor, *Encyclopedia of the Sciences of Learning*, pages 986–990. Springer US, Boston, MA, 2012.
44. S. Arnold, J. Fujima, and K. P. Jantke. Storyboarding serious games for large-scale training applications. In *Proc. CSEDU 2013*, pages 651–655. SciTePress, 2013.
45. S. Assar, I. Boughzala, and I. Boydens, editors. *Practical Studies in E-Government. Best Practices from Around the World*. Springer, 2011.
46. P. Atzeni and V. De Antonellis. *Relational database theory*. Addison-Wesley, Redwood City, 1993.
47. P. Atzeni, A. Gupta, and S. Sarawagi. Design and maintenance of data-intensive web-sites. In *Proceeding EDBT'98*, volume 1377 of *LNCS*, pages 436–450. Springer-Verlag, Berlin, 1998.
48. M. Augier and M. L. Frigotto. *Organization Theory*, pages 1–6. Palgrave Macmillan UK, London, 2016.
49. R. Aylett, S. Louchart, and A. Weallans. Research in interactive drama environments, role-play and story-telling. In *Proc. ICIDS 2011*, volume 7069 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2011.
50. C. W. Bachman. Data structure diagrams. In *Proc. IAG Conf. 'File 68' Helsingør*, pages 181–192, 1968. publ. No. 3; File design methods.
51. C. W. Bachman. Impact of object oriented thinking on ER modeling. In *Proc. ER'96*, volume 1157 of *Lecture Notes in Computer Science*, pages 1–4. Springer, 1996.

52. H.-J. Backe. Narrative rules? story logic and the structures of games. *LLC*, 27(3):243–260, 2012.
53. G. Bailly. *Techniques de menus : Caractérisation, Conception et Evaluation*. PhD thesis, Joseph Fourier University, Grenoble, France, 2009.
54. J. Banks. Semantics of simulation software. *Computer Integrated Manufacturing and Engineering*, 1:46 – 48, 1996.
55. L. Baresi, F. Garzotto, and P. Paolini. From web sites to web applications: New issues for conceptual modeling. In *ER Workshops 2000*, volume 1921 of *LNCS*, pages 89–100. Springer-Verlag, Berlin, 2000.
56. P. Barna, F. Frasincar, G.-J. Houben, and R. Vdovjak. Methodologies for web information system design. In *Proc. ITCC 2003*, pages 420–424, 2003.
57. L. Barroca, J. Hall, and P. Hall. *Software architectures - Advances and applications*. Springer, Berlin, 2000.
58. A. Bartoli, E. Medvet, and M. Mauri. Recording and Replaying Navigations on AJAX Web Sites. In *ICWE*, volume 7387 of *Lecture Notes in Computer Science*, pages 370–377, 2012.
59. R. Baskerville, J. Pries-Heje, and J. R. Venable. Soft design science methodology. In *Proc. DESRIST 2009*. ACM, 2009.
60. L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. Addison-Wesley, 2003.
61. C. Batini, S. Ceri, and S. Navathe. *Conceptual database design (an entity-relationship approach)*. Benjamin/Cummings, Redwood City, 1992.
62. S. Batsanov and A. Batsanov. *Introduction to Structural Chemistry*. Springer, 2012.
63. C. Beeri and B. Thalheim. Identification as a primitive of database models. In T. Polle, T. Ripke, and K.-D. Schewe, editors, *Proc. Fundamentals of Information Systems, 7th Int. Workshop on Foundations of Models and Languages for Data and Objects - FoMLaDO'98*, pages 19–36, Timmel, Ostfriesland, 1999. Kluwer, London.
64. J. H. T. Bekke. *Semantic data modelling*. Prentice-Hall, London, 1992.
65. I. P. Beletskaya, G. V. Latyshev, A. V. Tsvetkov, and N. V. Lukashev. The nickel-catalyzed SonogashiraHagihara reaction. *Tetrahedron Letters*, 44(27):5011–5013, June 2003.
66. J. Bell. Pragmatic reasoning: Inferring contexts. In *Proc. Context'1999*, LNAI 1688, pages 42–53. Springer, 1999.
67. L. Bellatreche. *Optimization and Tuning in Data Warehouses*, pages 1995–2003. Springer US, Boston, MA, 2009.
68. A. A. Benczúr and B. Kósa. Static analysis of structural recursion in semistructured databases and its consequences. In *Proc. ADBIS 2004*, volume 3255 of *Lecture Notes in Computer Science*, pages 189–203. Springer, 2004.
69. D. Benyon, T. R. G. Green, and D. Bentel. *Conceptual modeling for user interface development*. Practitioner series. Springer, 1999.
70. M. Berg. *Modelling of Natural Dialogues in the Context of Speech-based Information and Control Systems*. PhD thesis, Christian-Albrechts University, Computer Science Department, Kiel, 2014.
71. M. Berg, A. Düsterhöft, and B. Thalheim. Integration of natural language dialogues into the conceptual model of storyboard design. In *NLDB*, volume 6177 of *Lecture Notes in Computer Science*, pages 196–203. Springer, 2010.

72. M. Berg, A. Düsterhöft, and B. Thalheim. Query and answer forms for sophisticated database interfaces. In *Information Modelling and Knowledge Bases*, volume XXIV, pages 161–180. IOS Press, 2013.
73. M. Berg, B. Thalheim, and A. Düsterhöft. Integration of dialogue patterns into the conceptual model of storyboard design. In *ER Workshops*, volume 6413 of *Lecture Notes in Computer Science*, pages 160–169. Springer, 2010.
74. M. M. Berg. NADIA: A simplified approach towards the development of natural dialogue systems. In *Proc. NLDB 2015*, volume 9103 of *Lecture Notes in Computer Science*, pages 144–150. Springer, 2015.
75. A. N. Berger. The Economic Effects of Technological Progress: Evidence from the Banking Industry. *Journal of Money, Credit, and Banking*, 35(2):141 – 176, 2003.
76. R. Berghammer. Applying relation algebra and relview to solve problems on orders and lattices. *Acta Informatica*, 45(3):211–236, 2008.
77. M. Bergholtz, B. Andersson, and P. Johannesson. Towards a model of services based on co-creation, abstraction and restriction. In *Proc. 30th Int. Conf. ER 2011*, pages 107–116. Berlin, Heidelberg: Springer-Verlag, 2011.
78. N. O. Bernsen, L. Dybkjær, and H. Dybkjær. *Speech Interaction Theory*, pages 29–60. Springer London, London, 1998.
79. R. Berry. *Dramatis Personae*, pages 105–117. Palgrave Macmillan UK, London, 1993.
80. P. Bertolazzi, M. G. Fugini, and B. Pernici. Information system design based on reuse of conceptual components. In *Information Modeling in the New Millennium*, pages 219–230. IDEA group, 2001.
81. K. Bharat, T. Kamba, and M. Albers. Personalized, interactive news on the web. *Multimedia Systems*, 6(5):349–358, 1998.
82. T. Bhuiyan, Y. Xu, A. Jøsang, H. Liang, and C. Cox. Developing trust networks based on user tagging information for recommendation making. In *WISE*, pages 357–364, 2010.
83. M. Bichler, U. Frank, D. Avison, J. Malaurent, P. Fettke, D. Hovorka, J. Krämer, D. Schnurr, B. Müller, L. Suhl, and B. Thalheim. Theories in business and information systems engineering. *Business & Information Systems Engineering*, pages 1–29, 2016.
84. M. Bieber, F. Vitali, H. Ashman, V. Balasubramanian, and H. Oinas-Kukkonen. Fourth generation hypermedia: Some missing links for the world wide web. *Int. J. Human-Computer Studies*, 47:31–65, 1997.
85. A. Bienemann. *A generative approach to functionality of interactive information systems*. PhD thesis, CAU Kiel, Dept. of Computer Science, 2008.
86. A. Bienemann, K.-D. Schewe, and B. Thalheim. Towards a theory of genericity based on government and binding. In D. W. Embley, A. Olivé, and S. Ram, editors, *Conceptual Modeling - ER 2006, 25th International Conference on Conceptual Modeling, Tucson, AZ, USA, November 6-9, 2006, Proceedings*, volume 4215 of *Lecture Notes in Computer Science*, pages 311–324. Springer, 2006.
87. A. Binemann-Zdanowicz. Sitelang:edu: towards a context-driven e-learning content utilization model. In *Proc. ACM Symp. SAC*, pages 924–928. ACM, 2004.
88. A. Binemann-Zdanowicz. Towards generative engineering of content-intensive applications. In *Principles of Software Engineering Conference (PRISE 2004)*, pages 41–49, 2004.

89. A. Binemann-Zdanowicz, R. Kaschek, K.-D. Schewe, and B. Thalheim. Context-aware web information systems. In S. Hartmann and J. Roddick, editors, *Conceptual Modelling 2004 – First Asia-Pacific Conference on Conceptual Modelling (APCCM 2004)*, volume 31 of *CRPIT*, pages 37–48, Dunedin, New Zealand, 2004. Australian Computer Society.
90. A. Binemann-Zdanowicz, K.-D. Schewe, and B. Thalheim. Adaptation to learning styles. In Kinshuk, C.-K. Looi, E. Sutinen, D. G. Sampson, I. Aedo, L. Uden, and E. Kähkönen, editors, *Proceedings of the IEEE International Conference on Advanced Learning Technologies (ICALT 2004)*, pages 121–125. IEEE Computer Society, 2004.
91. A. Binemann-Zdanowicz, K.-D. Schewe, and B. Thalheim. Development of collaboration frameworks for distributed web information systems. In G. Kotisis, D. Taniar, S. Bressan, I. K. Ibrahim, and S. Mokhtar, editors, *iiWAS'2005 - The Seventh International Conference on Information Integrationand Web-based Applications Services*, volume 196 of *books@ocg.at*, pages 551–562. Austrian Computer Society, 2005.
92. A. Binemann-Zdanowicz, B. Schulz-Brünken, B. Thalheim, and B. Tschedel. Flexible e-payment based on content and profile in the e-learning system DaMiT. In *Proc. E-Learn 2003*, pages 16–27. Association for the Advancement of Computing in Education, 2003.
93. A. Binemann-Zdanowicz, B. Thalheim, K.-D. Schewe, and J. Zhao. Quality assurance in the design of web information systems. In *Fifth International Conference on Quality Software (QSIC 2005)*, pages 91–98. IEEE Computer Society, 2005.
94. A. Binemann-Zdanowicz, B. Thalheim, and B. Tschedel. Logistics for learning objects. In *Proceedings of eTrain 2003*, 2003.
95. J. Bisbal, D. Lawless, B. Wu, and J. Grimson. Legacy Information Systems: Issues and Directions. *IEEE Software*, 16:103–111, 1999.
96. J. Biskup. *Security in Computing Systems - Challenges, Approaches and Solutions*. Springer, 2009.
97. D. Bjørner. *Software Engineering 3: Domains, requirements, and software design*. Springer, Berlin, 2006.
98. D. Bjørner. *Domain engineering*, volume 4 of *COE Research Monographs*. Japan Advanced Institute of Science and Technology Press, Ishikawa, 2009.
99. M. Black. *Models and Metaphors*. Cornell University Press, Ithaca, 1962.
100. C. Bobed, R. Trillo, E. Mena, and S. Ilarri. From keywords to queries: Discovering the user's intended meaning. In *WISE*, pages 190–203, 2010.
101. M. A. Bochicchio and N. Fiore. WARP: web application rapid prototyping. In *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC), Nicosia, Cyprus, March 14-17, 2004*, pages 1670–1676. ACM, 2004.
102. M. A. Bochicchio and A. Longo. Conceptual modeling of data intensive and information intensive web applications. In *Proc. MMM 2004*, pages 292–299. IEEE Computer Society, 2004.
103. B. Boehm. A view of 20th and 21st century software engineering. In *Proc. ICSE'06*, pages 12–29. ACM Press, 2006.
104. B. Boiko. *Content Management Bible*. Wiley, Indianapolis, 2001.
105. M. Boman, J. B. Jr., P. Johannesson, and B. Wangler. *Methodenneutraler Fachentwurf*. Prentice Hall, London, 1996.

106. A. Bonifati, S. Ceri, P. Fraternali, and A. Maurino. Building multi-device, content-centric applications using WebML and the W3I3 tool suite. In *ER Workshops 2000*, volume 1921 of *LNCS*, pages 64–75. Springer-Verlag, Berlin, 2000.
107. P. Bonnet and D. Shasha. *Schema Tuning*, pages 2497–2499. Springer US, Boston, MA, 2009.
108. E. Börger. *Computability, Complexity, Logic*, volume 128 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1989.
109. E. Börger. The ASM refinement method. *Formal Aspects of Computing*, 15:237–257, 2003.
110. E. Börger and R. Stärk. *Abstract State Machines*. Springer-Verlag, Berlin Heidelberg New York, 2003.
111. A. Borgida, V. K. Chaudhri, P. Giorgini, and E. S. K. Yu, editors. *Conceptual Modeling: Foundations and Applications - Essays in Honor of John Mylopoulos*, volume 5600 of *Lecture Notes in Computer Science*. Springer, 2009.
112. P. Boström, M. Neovius, I. Oliver, and M. A. Waldén. Formal transformation of platform independent models into platform specific models. In *B 2007, Proceedings*, volume 4355 of *Lecture Notes in Computer Science*, pages 186–200. Springer, 2007.
113. I. Boughzala, M. Janssen, and S. Assar, editors. *Case Studies in e-Government 2.0. Changing Citizen Relationships*. Springer, 2015.
114. P. Bouquet, L. Serafini, P. Brezillon, M. Benerecetti, and F. Castellani, editors. *Modeling and Using Context – Proceedings of CONTEXT'99*, volume 1688 of *LNAI*. Springer-Verlag, Berlin, 1999.
115. P. Bradford. *Information architects*. Graphics US Inc., 1995.
116. M. Brambilla and P. Fraternali. Large-scale model-driven engineering of web user interaction: The webml and webratio experience. *Sci. Comput. Program.*, 89:71–87, 2014.
117. M. Bramer. *Web Programming with PHP and MySQL - A Practical Guide*. Springer, 2015.
118. V. Breazu-Tannen, P. Buneman, and S. A. Naqvi. Structural recursion as a query language. In *Database Programming Languages: Bulk Types and Persistent Data*, pages 9–19. Morgan Kaufmann, 1991.
119. M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, editors. *On conceptual modelling (Perspectives from artificial intelligence, databases, and programming languages)*. Springer, New York/Berlin, 1984.
120. M. L. Brodie and M. Stonebraker. *Migrating Legacy Systems: Gateways, Interfaces, and the Incremental Approach*. Morgan Kaufmann, 1995.
121. S. Browers, L. Delcambre, and D. Maier. Superimposed schematics: Introducing E-R structure for in-situ information selections. In *Proc. ER'02*, volume LNCS 2503, pages 90–104, Berlin, 2002. Springer.
122. L. Brown. *Integration Models: Templates for Business Transformation - The Authoritative Solution*. Sams Publishing, 2000.
123. G. Bruinsma and D. Weisburd, editors. *Life Events*, pages 2937–2937. Springer New York, New York, NY, 2014.
124. P. Brusilovsky, A. Kobsa, and W. Nejdl, editors. *The Adaptive Web - Methods and Strategies of Web Personalization*. Springer, 2007.
125. P. Brusilovsky. Adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 11(1):87–110, 2001.

126. D. Buchholz, H. Cyriaks, A. Düsterhöft, H. Mehlan, and B. Thalheim. Applying a natural language dialogue tool for designing databases. In M. Bouzeghoub, editor, *Proc. NLDB'95*, pages 119–133, Paris, 1995. Afct.
127. E. Buchholz, A. Düsterhöft, and B. Thalheim. Capturing information on behaviour with the RADD-NLI: A linguistic and knowledge-based approach. *Data Knowl. Eng.*, 23(1):33–46, 1997.
128. K. Bühler. *Sprachtheorie: Die Darstellungsfunktion der Sprache*. Fischer, Stuttgart, 1965.
129. M. Busch and N. Koch. MagicUWE - A CASE tool plugin for modeling web applications. In *Proc. ICWE 2009*, volume 5648 of *Lecture Notes in Computer Science*, pages 505–508. Springer, 2009.
130. H. Bußmann. *Lexikon der Sprachwissenschaft*. Kröner, Stuttgart, 1990.
131. J. Buur and J. Windum. *Man machine interface design*. Danish Design Centre, Copenhagen, 1994.
132. B. Buxton. *Sketching User Experiences getting the design right and the right design*. Norman Kaufmann Publishers, San Francisco, 2007.
133. D. Calvisi. *Story maps: How to Write a Great Screenplay*. Act Four Screenplays, 2012.
134. J. Campbell. *The Hero with a Thousand Faces*. New World Library, Novato, 2008.
135. U. Cantner, E. Dinopoulos, H. Hanusch, and L. Orsenigo, editors. *Journal of Evolutionary Economics*, volume 1–26. Springer, 1991.
136. J. V. Carlis, S. T. March, and G. W. Dickson. Physical database design: A DSS approach. *Information and Management*, 6(4):211–224, 1983.
137. D. Carlson. *Modeling XML Applications with UML: Practical e-Business Applications*. Addison-Wesley Object Technology Series, 2001.
138. D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. B. Zdonik. Monitoring streams - A new class of data management applications. In *VLDB*, pages 215–226. Morgan Kaufmann, 2002.
139. C. Carreras, J. C. López, M. L. L. Vallejo, L. Sánchez, C. D. Kloos, and N. M. Madrid. A co-design methodology based on formal specification and high-level estimation. In *Proc. CODES*, pages 28–35. IEEE Computer Society, 1996.
140. J. M. Carroll, editor. *Designing Interaction: Psychology at the Human-Computer Interface*. Cambridge University Press, Cambridge, England, 1991.
141. J. M. Carroll, editor. *Making use: scenario-based design of human-computer interactions*. MIT Press, Cambridge, England, 2000.
142. J. Caumanns. *Automatisierte Komposition von wissensvermittelnden Dokumenten für das World Wide Web*. PhD thesis, Brandenburg University of Technology at Cottbus, Faculty of Mathematics, Natural Sciences and Computer Science, 2000.
143. S. Ceri and P. Fraternali. *Designing Database Applications with Objects and Rules : The IDEA Methodology*. Addison-Wesley, Boston, MA, USA, 1st edition, 1997.
144. S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications*. Morgan Kaufmann, San Francisco, 2003.
145. G. Chen and W. Starosta. *Foundations of Intercultural Communication*. Allyn and Bacon, Boston, 1998.

146. P. P. Chen. The entity-relationship model: Toward a unified view of data. *ACM TODS*, 1(1):9–36, 1976.
147. P. P. Chen, B. Thalheim, and L. Y. Wong. Future directions of conceptual modeling. In *Conceptual Modeling, Current Issues and Future Directions, Selected Papers from the Symposium on Conceptual Modeling, Los Angeles, California, USA, held before ER'97*, LNCS 1565, pages 287–301. Springer, Berlin, 1999.
148. N. Chomsky. *Some concepts and consequences of the theory of government and binding*. MIT Press, 1982.
149. N. Chomsky. *Lectures on government and binding - The Pisa lectures*. Mouton, De Gryuter, 1993.
150. A. Choungourian. Color preferences and cultural variation. In *Perceptual and Motor Skills*, volume 26, pages 1203 – 1206. Southern Universities Press, 1968.
151. J. Chu-Carroll. Form-based reasoning for mixed-initiative dialogue management in information-query systems. In *Proc. EUROSPEECH'99*. ISCA, 1999.
152. S. Cilella, C. Berman, and J. Rheinfrank. Experience definition through storyboarding. In *Proc. 4th Int. Conf. on Tangible and Embedded Interaction*, pages 325–328. ACM, 2010.
153. W. Clauß. Using structural recursion as query mechanism for data models with references. In *Proc. ER'96*, volume 1157 of *Lecture Notes in Computer Science*, pages 134–145. Springer, 1996.
154. W. Clauß and B. Thalheim. Abstraction layered structure-process codesign. In *Proc. 8th Int. Conf. COMAD 1997*, pages 223–237, Chennai (Madras), India, 1998. Narosa Publishing, New Delhi.
155. CMMI. Capability maturity model integration, version 1.2. CMMI for development. Technical report, CMU/SEI-2006-TR-008, August 2006.
156. S. Comella-Dorda, K. Wallnau, R. Seacord, and J. Robert. A Survey of Black-Box Modernization Approaches for Information Systems. In *ICSM '00: Proceedings of the International Conference on Software Maintenance (ICSM'00)*, page 173, Washington, DC, USA, 2000. IEEE Computer Society.
157. J. Conallen. Modeling web application architectures with UML. *Commun. ACM*, 42(10):63–70, 1999.
158. J. Conallen. *Building Web Applications with UML*. Addison-Wesley, Boston, 2003.
159. J. H. Connolly. Context in the study of human languages and computer programming languages: A comparison. In *Proc. Context'2001*, LNAI 2116, pages 116–128. Springer, 2001.
160. T. Connolly and C. Begg. *Database solutions - A step-by-step guide to building databases*. Pearson, Harlow, 2004.
161. J. O. Coplien and D. C. Schmidt, editors. *Pattern languages for program design*. Addison-Wesley, Reading, 1995.
162. CoreMedia. Homepage of the CoreMedia AG. <http://www.coremedia.com>, 2016.
163. C. Courage and K. Baxter. *Understanding your users: a practical guide to user requirements - methods, tools & techniques*. Morgan Kaufman, Boston, 2005.
164. J. F. Cragan and D. C. Shields. *Understanding Communication Theory*. Allyn and Bacon, 1998.
165. A. Cruse. *Meaning in Language: An Introduction to Semantics and Pragmatics*. Oxford University Press, Oxford, 2000.
166. D. Crystal. *The Cambridge encyclopedia of language*. Cambridge University Press, 1987.

167. M. J. E. Cuaresma and G. Aragón. NDT. A model-driven approach for web requirements. *IEEE Trans. Software Eng.*, 34(3):377–390, 2008.
168. M. J. E. Cuaresma and N. Koch. Requirements engineering for web applications - A comparative study. *J. Web Eng.*, 2(3):193–212, 2004.
169. M. J. E. Cuaresma and N. Koch. Metamodeling the requirements of web systems. In *WEBIST 2005 and WEBIST 2006. Revised Selected Papers*, volume 1 of *Lecture Notes in Business Information Processing*, pages 267–280. Springer, 2007.
170. K. Czarnecki and U. W. Eisenecker. *Generative programming - methods, tools and applications*. Addison-Wesley, 2000.
171. M. Dadashzadeh and D. W. Stemple. Converting SQL queries into relational algebra. *Information & Management*, 19(5):307–323, 1990.
172. A. Dahanayake and B. Thalheim. Co-evolution of (information) system models. In *Enterprise, Business-Process and Information Systems Modeling*, number 50 in LNBIB, pages 314–326. Springer, 2010.
173. A. Dahanayake and B. Thalheim. Towards a framework for emergent modeling. In *ER Workshops*, volume 6413 of *Lecture Notes in Computer Science*, pages 128–137. Springer, 2010.
174. A. Dahanayake and B. Thalheim. A conceptual model for IT service systems. *Journal of Universal Computer Science*, 18(17):2452–2473, 2012.
175. A. Dahanayake and B. Thalheim. *Correct Software in Web Applications and Web Services*, chapter W*H: The conceptual Model for services, pages 145–176. Texts & Monographs in Symbolic Computation. Springer, Wien, 2015.
176. DAMA. Drm bibliography. <https://www.dama.org/content/drm-bibliography>, accessed Sept. 28, 2016, 2016.
177. DaMiT. The intelligent data mining tutorial workbench. <http://DaMiT.dfki.de>, 2005.
178. M. Danesi. The neurological coordinates of metaphor. *Communication & Cognition*, 22(1):73–86, 1989.
179. G. S. Danford and B. Tauke. Universal design New York. State University of New York at Buffalo. Center for Inclusive Design & Environmental Access and New York (N.Y.). Mayor's Office for People with Disabilities and New York (N.Y.). Dept. of Design and Construction and American Institute of Architects. New York Chapter and New York (N.Y.). Office of the Mayor, 2001.
180. R. Darimont and A. van Lamsweerde. Formal refinement patterns for goal-driven requirements elaboration. In *SIGSOFT FSE*, pages 179–190, 1996.
181. C. Date. *Database in depth: Relational theory for practitioners*. O'Reilly, Sebastopol, 2005.
182. O. De Troyer, S. Casteleyn, and P. Plessers. WSDM: Web semantics design method. In *Web Engineering: Modelling and Implementing Web Applications*, pages 303–351. Springer-Verlag London Limited, 2008.
183. O. De Troyer and C. Leune. WSDM: A user centered design method for web sites. *Computer Networks*, 30:85–94, 1998.
184. O. De Troyer and C. Leune. WSDM: A user-centered design method for web sites. In *Computer Networks and ISDN Systems – Proceedings of the 7th International WWW Conference*, pages 85–94. Elsevier, 1998.
185. M. C. DeAngelis. *Data Modeling with ERwin*. Sams Publishing, Indiana, 2000.
186. B. Debatin. Literature on the theory of metaphor. <http://www.uni-leipzig.de/~debatin/english/Research/Metaphor.htm>.

187. B. Debatin. Metaphors and computers. *Semiotic Review of Books*, 3(1), 1991.
188. W. Deiters, T. Löffeler, and S. Pfennigschmidt. *The Information Logistics Approach Toward User Demand-Driven Information Supply*, pages 37–48. Springer US, Boston, MA, 2003.
189. Y. Dennebouy, M. Andersson, A. Auddino, Y. Dupont, E. Fontana, M. Gentile, and S. Spaccapietra. SUPER: Visual interfaces for object+relationship data models. *Journal of Visual Languages and Computing*, 6(1):73–99, 1995.
190. N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3(1/2):69–116, 1987.
191. D. Dori. *Object-process methodology: A holistic system paradigm*. Springer, Heidelberg, 2002.
192. D. Dori. Object-process methodology for structure-behavior co-design. In *Handbook of Conceptual Modeling - Theory, Practice, and Research Challenges*, pages 209–258. Springer, 2011.
193. G. Dowek. *Principles of Programming Languages*. Pearson, 2009.
194. T. Downey. *Guide to Web Development with Java - Understanding Website Creation*. Springer, 2012.
195. D. Draheim and G. Weber. Modelling form-based interfaces with bipartite state machines. *Interacting with Computers*, 17(2):207–228, 2005.
196. D. Draheim and G. Weber. *Form-Oriented Analysis -A New Methodology to Model Form-Based Applications*. Springer Verlag, 2010.
197. A. Dresch, D. Pacheco Lacerda, and J. A. Valle Antunes Jr., editors. *Design Science Research: A Method for Science and Technology Advancement*. Springer, 2015.
198. H. Dugas. *Bioorganic Chemistry*. Springer, New York, 1998.
199. E. B. Duncan. A faceted approach to hypertext? In *UK Hypertext*, pages 157–163, 1988.
200. A. Düsterhöft. *Zur natürlichsprachlichen interaktiven Unterstützung im Datenbankentwurf*, volume 36 of *DISDBIS*. Infix Verlag, St. Augustin, Germany, 1997.
201. A. Düsterhöft and K.-D. Schewe. Conceptual modeling of application stories. In *The Handbook of Conceptual Modeling: Its Usage and Its Challenges*, chapter 11, pages 359–377. Springer, Berlin, 2011.
202. A. Düsterhöft and B. Thalheim. Integrating retrieval functionality in websites based on storyboard design and word fields. In *Proc. NLDB'02*, volume 2553 of *LNCS*, pages 52–63. Springer, 2002.
203. A. Düsterhöft and B. Thalheim. *Information Modeling for Internet applications*, chapter Systematic development of internet sites: Extending approaches of conceptual modeling, pages 80–101. Idea Group Publishing, 2003.
204. A. Düsterhöft and B. Thalheim. Linguistic based search facilities in snowflake-like database schemes. *Data and Knowledge Engineering*, 48:177–198, 2004.
205. M. Duzi, A. Heimbürger, T. Tokuda, P. Vojtas, and N. Yoshida. Multi-agent knowledge modelling. In *Information Modelling and Knowledge Bases*, volume XX, pages 411–428, Amsterdam, 2009. IOS Press.
206. R. E. Eberts. *User interface design*. Prentice Hall, Englewood Cliffs, 1994.
207. H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors. *Handbook of graph grammars and computing by graph transformations. Vol. 2: Applications, languages and tools*. World Scientific, Singapore, 1999.
208. D. Eick. *Drehbuchtheorien: Ein vergleichende Analyse*. UVK Verlagsge-sellschaft, Konstanz, 2006.

209. R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems*. Addison-Wesley, Boston, MA, USA, 4th edition, 2004.
210. D. Embley and B. Thalheim, editors. *Handbook of conceptual modelling: Its Usage and Its Challenges*. Springer, 2011.
211. D. W. Embley. *Object database development - concepts and principles*. Addison-Wesley, 1998.
212. L. M. Encarna o. Adaptivity in graphical user interfaces: an experimental framework. *Computers & Graphics*, 19(6):873–884, 1995.
213. I. ERwin. ERwin data modeler. <http://erwin.com/products/data-modeler>, accessed Sept. 28, 2016, 2016.
214. W. Esswein, S. Lehrmann, and H. Schlieter. Referenzmodelle. *Das Wirtschaftsstudium*, 39(3):371–375, 2010.
215. E. Falkenberg, W. Hesse, P. Lindgreen, B. Nilsson, J. Oei, C. Roland, R. Stamper, F. V. Assche, A. Verijn-Stuart, and K. Voss. FRISCO - A framework for information systems concepts. Technical report, IFIP Working Group 8.1, Task group FRISCO, 1998. <http://www.mathematik.uni-marburg.de/~hesse/papers/fri-full.pdf>.
216. D. Fensel, F. M. Facca, E. P. B. Simperl, and I. Toma. *Semantic Web Services*. Springer, 2011.
217. T. Fernandes. *Global interface design*. Addison-Wesley, Boston, 1995.
218. A. Fernandez, E. Insfr n, and S. M. Abrah o. Integrating a usability model into model-driven web development processes. In *WISE*, pages 497–510, 2009.
219. S. Fertig, E. Freeman, and D. Gelernter. Lifestreams: An alternative to the desktop metaphor, 1996.
220. P. Fettke and P. Loos. *Reference Modeling for Business Systems Analysis*, volume 1, chapter Perspectives on Reference Modeling, pages 1–21. Capítulo, 2007.
221. T. Feyer, O. Kao, K.-D. Schewe, and B. Thalheim. Design of data-intesive web-based information services. In Q. Li, Z. M.  zsoyoglu, R. Wagner, Y. Kamabayashi, and Y. Zhang, editors, *Proceedings of the First International Conference on Web Information Systems Engineering, Volume I (WISE 2000)*, pages 462–467. IEEE Computer Society, 2000.
222. T. Feyer, K.-D. Schewe, and B. Thalheim. Conceptual design and development of information services. In T. W. Ling, S. Ram, and M.-L. Lee, editors, *Conceptual Modeling - ER '98, 17th International Conference on Conceptual Modeling*, volume 1507 of *Lecture Notes in Computer Science*, pages 7–20. Springer, 1998.
223. G. Fiedler, A. Czerniak, D. Fleischer, H. Rumohr, M. Spindler, and B. Thalheim. Content Warehouses. Preprint 0605, Department of Computer Science, Kiel University, 2006.
224. G. Fiedler, H. Jaakkola, T. M kinen, B. Thalheim, and T. Varkoi. Application domain engineering for web information systems supported by SPICE. In *Proc. SPICE'07*, Seoul, Korea, May 2007. IOS Press.
225. G. Fiedler, H. Jaakkola, T. M kinen, B. Thalheim, and T. Varkoi. Co-design of web information systems supported by SPICE. In *Information Modelling and Knowledge Bases*, volume XX, pages 123–138, Amsterdam, 2009. IOS Press.
226. G. Fiedler, U. Krautz, and B. Thalheim. SeSAM - support on profile, portfolio, and demand for (e-)parliamentarians. In *Proc. e-Society'04*, 2004.
227. G. Fiedler and B. Thalheim. Towards linguistic foundations of content management. In Springer, editor, *NLDB'2004*, LNCS 3136, pages 348–353, 2004.

228. S. Field. *Screenplay: The Foundations of Screenwriting*. Delta, 2005.
229. C. Fiell and P. Fiell. *Design handbook*. Taschen, London, 2006.
230. G. Fischer. User modeling in human-computer interaction. *User Model. User-Adapt. Interact.*, 11(1-2):65–86, 2001.
231. G. Fitzpatrick, W. Tolone, S. Kaplan, and M. Work. Local and distributed distributed social world. In *ECSW*, pages 1–16, 1995.
232. D. Flanagan. *JavaScript - The Definitive Guide: Activate Your Web Pages: Covers ECMAScript 5 and HTML5* (6. ed.). O'Reilly, 2011.
233. D. Flanagan. *JavaScript - Pocket Reference: Activate Your Web Pages, Third Edition*. O'Reilly, 2012.
234. D. Fleischer and K. Jannaschek. A path to filled archives. *Nature Geoscience*, pages 575 – 576, 2011.
235. C. C. Fleming and B. von Halle. *Handbook of relational database design*. Addison-Wesley, Reading, MA, 1989.
236. J. Fleming. *Web Navigation - Designing the user experience*. O'Reilly, Sebastopol, 1998.
237. G. Fliedl, C. Kop, W. Mayerthaler, H. C. Mayr, and C. Winkler. The NIBA approach to quantity settings and conceptual predesign. In *Proc. NLDB'01*, volume 3 of *LNI*, pages 211–214. GI, 2001.
238. G. Fliedl, C. Kop, and H. C. Mayr. From scenarios to KCPM dynamic schemas: Aspects of automatic mapping. In *Proc. NLDB'03*, volume 29 of *LNI*, pages 91–105. GI, 2003.
239. G. Fliedl, C. Kop, and H. C. Mayr. From textual scenarios to a conceptual schema. *Data Knowl. Eng.*, 55(1):20–37, 2005.
240. G. Fliedl, C. Kop, H. C. Mayr, W. Mayerthaler, and C. Winkler. Linguistically based requirements engineering - the niba-project. *Data Knowl. Eng.*, 35(2):111–120, 2000.
241. W. Fokkink and H. Zantema. Basic process algebra with iteration: Completeness of its equational axioms. *The Computer Journal*, 37(4):259–268, 1994.
242. W. Fokkink and H. Zantema. Termination modulo equations by abstract commutation with an application to iteration. *Theoretical Computer Science*, 177(2):407–423, 1997.
243. H. I. Forsha. *The Complete Guide to Storyboarding and Problem Solving*. ASQ Quality Press, 1994.
244. F. Förster and B. Thalheim. An effectual approach for a data and information management for humanists. *Qualitative and Quantitative Methods in Libraries (QQML)*, 2:121–128, 2012.
245. S. Fowler and V. Stanwick. *The GUI style guide*. Academic Press, Amsterdam, 1995.
246. X. Franch, A. Maté, J. Trujillo, and C. Cares. On the joint use of i^* with other modelling frameworks: A vision paper. In *Proc. RE 2011*, pages 133–142. IEEE Computer Society, 2011.
247. S. Francia. *MongoDB and PHP - Document-Oriented Data for Web Developers*. O'Reilly, 2012.
248. U. Frank and S. Strecker. Open reference models - community-driven collaboration to promote development and dissemination of reference models. *Enterprise Modelling and Information Systems Architectures*, 2(2):32–41, 2007.
249. F. Frasincar. *Hypermedia Presentation Generation for Semantic Web Information Systems*. University Press Facilities, 2005.

250. F. Frasincar, P. Barna, G. Houben, and Z. Fiala. Adaptation and reuse in designing web information systems. In *Proc. ITCC'04*, pages 387–291. IEEE Computer Society, 2004.
251. F. Frasincar, G.-J. Houben, and R. Vdovjak. An RMM-based methodology for hypermedia presentation design. In *ADBIS 2001, Vilnius, Lithuania*, volume 2151 of *Lecture Notes in Computer Science*, pages 323–337. Springer, 2001.
252. P. Fraternali and S. Ceri. *Designing Database Applications With Objects and Rules: The Idea Methodology*. Series on Database Systems and Applications. Addison-Wesley, 1997.
253. P. Fraternali and P. Paolini. Model-driven development of web applications: the AutoWeb system. *ACM Trans. Inf. Syst.*, 18(4):323–382, 2000.
254. S. Friedemann, K. P. Jantke, and L. Baumbach. Textbook gamification: Methods and technologies. In *Proc. CSEDU 2015, Revised Selected Papers*, volume 583 of *Communications in Computer and Information Science*, pages 406–424. Springer, 2015.
255. G. Frisch. *Using open source software to develop E-business applications for medium- and small-sized enterprises*. PhD thesis, Bratislava, Univ., 2009.
256. J. Fujima, K. P. Jantke, and O. Arnold. Media multiplicity at your fingertips: Direct manipulation based on webbles. In *2012 IEEE International Conference on Multimedia and Expo Workshops*, pages 217–222. IEEE, 2012.
257. J. Fujima, K. P. Jantke, and S. Arnold. Digital game playing as storyboard interpretation. In *Proc. IGIC 2013*, pages 64–71. IEEE, 2013.
258. Y. Fukazawa, T. Naganuma, K. Fujii, and S. Kurakake. *Role-Task Model for Advanced Task-Based Service Navigation System*, pages 1021–1028. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.
259. H. Furusawa. A free construction of kleene algebras with tests. In D. Kozen, editor, *Proc. MPC 2004*, pages 129–141. Springer, Berlin, 2004.
260. M. Gädke and K. Turowski. Generic web-based federation of business application systems for e-commerce applications. In *EFIS 1999*, pages 25–42. Infix Verlag, 1999.
261. B. R. Gaines. Designing visual languages for description logics. *Journal of Logic, Language and Information*, 18(2):217–250, 2009.
262. P. Galand, F. Hallyn, C. Lvy, and W. Verbaal, editors. *Quintilien ancien et moderne. Etudes runies*. Brepols Publishers, 2010.
263. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.
264. B. Ganter and R. Wille. *Formal concept analysis - Mathematical foundations*. Springer, Berlin, 1998.
265. J. A. García-García, M. A. Ortega, L. García-Borgoñón, and M. J. Escalona. Ndt-suite: A model-based suite for the application of NDT. In *Proc. ICWE 2012*, volume 7387 of *Lecture Notes in Computer Science*, pages 469–472. Springer, 2012.
266. J. Garrett. *The Elements of User Experience: User-centered Design for the Web*. New Riders Press, Indiana, 2002.
267. I. Garrigós, J.-N. Mazón, and J. Trujillo. A requirement analysis approach for using i* in web engineering. In *ICWE*, volume 5648 of *Lecture Notes in Computer Science*, pages 151–165, 2009.

268. F. Garzotto, P. Paolini, D. Bolchini, and S. Valenti. “Modeling-by-Patterns” of web applications. In *Proceedings of WWWCM’99, Lecture Notes in Computer Science*, 1727, pages 293–306, Paris, France, December 1999. Springer.
269. F. Garzotto, P. Paolini, and D. Schwabe. HDM - a model-based approach to hypertext application design. *ACM ToIS*, 11(1):1–26, 1993.
270. S. E. Gaudl, K. P. Jantke, and C. Woelfert. The good, the bad and the ugly: Short stories in short game play. In *Proc. ICIDS 2009*, volume 5915 of *Lecture Notes in Computer Science*, pages 127–133. Springer, 2009.
271. H.-W. Gellersen and M. Gaedke. Object-oriented web application development. *IEEE Internet Computing*, 3(1):60–68, 1999.
272. M. R. Genesereth and N. J. Nilsson. *Logical foundations of artificial intelligence*. Morgan-Kaufman, Los Altos, 1988.
273. D. E. Gentner and J. Grudin. Design models for computer-human interfaces. *Computer*, 6:28–35, 1996.
274. P. Germanakos and M. Belk. *User Modeling*, pages 79–102. Springer International Publ., Cham, 2016.
275. P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani. Reasoning with goal models. In *ER*, pages 167–181, 2002.
276. C. Glodt, P. Kelsen, N. Amálio, and Q. Ma. From platform-independent to platform-specific models using democles. In *Proc. 24th Annual ACM SIGPLAN*, pages 795–796. ACM, 2009.
277. P. Gloor. *Elements of hypermedia design: Techniques for navigation & visualization in cyberspace*. Birkhauser, Boston, 1996.
278. J. W. Goethe. *Farbenlehre*. Cotta, Stuttgart, 1810.
279. M. Gogolla. *An extended entity-relationship model - fundamentals and pragmatics*. LNCS 767. Springer, Berlin, 1994.
280. C. Goldfarb. *The SGML handbook*. OUP, Oxford, 1990.
281. D. Goldin, S. Srinivasa, and B. Thalheim. IS = DBS + interaction - towards principles of information systems. In *Proc. 19th. Int. Conf. ER’2000*, pages 140–153. Berlin, Heidelberg: Springer-Verlag, 2010.
282. J. Gómez, C. Cachero, and O. Pastor. Conceptual modeling of device-independent web applications. *IEEE MultiMedia*, 8(2):26–39, 2001.
283. M. González, L. Cernuzzi, N. Aquino, and O. Pastor. Developing web applications for different architectures: The MoWebA approach. In *Proc. RCIS 2016*, pages 1–11. IEEE, 2016.
284. M. González, L. Cernuzzi, and O. Pastor. A navigational role-centric model oriented web approach - moweba. *Int. J. Web Eng. Technol.*, 11(1):29–67, 2016.
285. D. S. Gorfein and W. G. Stone. Encoding interval in short-term memory. *Journal of Verbal Learning and Verbal Behavior*, 6:520 – 522, 1967.
286. A. Graesser, V. Rus, and Z. Cai. Question classification schemes. In *Proc. Workshop on the Question Generation Shared Task and Evaluation Challenge*, pages 8–9, 2008.
287. M. Green. *Report on Dialogue Specification Tools*. In: G. E. Pfaff (Ed.) “User Interface Management Systems”. Springer Verlag, Berlin, 1985.
288. S. Greenberg, M. S. T. Carpendale, N. Marquardt, and B. Buxton. The narrative storyboard: telling a story about use and context over time. *Interactions*, 19(1):64–69, 2012.

289. I. Griffiths, M. Adams, and J. Liberty. *Programming C# 4.0 - Building Windows, Web, and RIA Applications for .NET with C# 4.0: Visual Studio 2010, .NET 4, and more* (6. ed.). O'Reilly, 2010.
290. J. Grigera, J. M. Rivero, E. R. Luna, F. Giacosa, and G. Rossi. From requirements to web applications in an agile model-driven approach. In *ICWE*, volume 7387 of *Lecture Notes in Computer Science*, pages 200–214, 2012.
291. H. Gudjons. *Pädagogisches Grundwissen*. Klinkhardt, Bad Heilbrunn, 2001.
292. N. Güell, D. Schwabe, and P. Vilain. Modeling interactions and navigation in web applications. In S. W. Liddle, H. C. Mayr, and B. Thalheim, editors, *Conceptual Modeling for E-Business and the Web*, volume 1921 of *LNCS*, pages 115–127. Springer-Verlag, 2000.
293. Y. Guo and D. H.-L. Goh. From storyboard to software: user evaluation of an information literacy game. In *Proc. 31st Ann. ACM Symposium on Applied Computing*, pages 199–201. ACM, 2016.
294. M. Hadjouni, H. B. Zghal, M.-A. Aufaure, and H. B. Ghézala. User modeling-based spatial web personalization. In *KES (2)*, volume 6882 of *Lecture Notes in Computer Science*, pages 41–50, 2011.
295. M. Haesen, K. Luyten, and K. Coninx. Get your requirements straight: Storyboarding revisited. In *Proc. INTERACT 2009, Part II*, volume 5727 of *Lecture Notes in Computer Science*, pages 546–549. Springer, 2009.
296. M. Haesen, J. Van den Bergh, J. Meskens, K. Luyten, S. Degrandart, S. Demeyer, and K. Coninx. *Using Storyboards to Integrate Models and Informal Design Knowledge*, pages 87–106. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
297. D. M. Haftor, M. Kajtazi, and A. Mirijamdotter. *A Review of Information Logistics Research Publications*, pages 244–255. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
298. E. Hall. *Beyond Culture*, volume 222 (13). Anchor Books, New York, 1976.
299. T. A. Halpin. *Information Modeling and Relational Databases*. Morgan Kaufmann Publishers, 2001.
300. T. A. Halpin. Comparing metamodels for ER, ORM and UML data models. In *Advanced Topics in Database Research*, volume 3, pages 23–44, 2004.
301. S. H. Han and S. W. Hong. A systematic approach for coupling user satisfaction with product design. *Ergonomics*, 46(13-14):1441–1461, 2003.
302. T. Härdter. DBMS architecture - still an open problem. In *BTW*, volume 65 of *LNI*, pages 2–28. GI, 2005.
303. T. Härdter. XML databases and beyond - plenty of architectural challenges ahead. In *ADBIS*, volume 3631 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2005.
304. D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.
305. D. Harel and R. Marelly. *Come, Let's play: Scenario-based programming using LSCs and the play-engine*. Springer, Berlin, 2003.
306. D. Harel and A. Naamad. The STATEMATE semantics of statecharts. *ACM Transactions on Software Engineering and Methodology*, 5(4):293–333, 1996.
307. S. Hartmann. Reasoning about participation constraints and Chen's constraints. In *ADC*, volume 17 of *CRPIT*, pages 105–113. Australian Computer Society, 2003.

308. S. Hartmann, A. Hoffmann, S. Link, and K.-D. Schewe. Axiomatizing functional dependencies in the higher-order entity-relationship model. *Inf. Process. Lett.*, 87(3):133–137, 2003.
309. R. Hausser. *Foundations of Computational Linguistics*. Springer-Verlag, Berlin, Germany, 2001.
310. R. Hausser. *A computational model of natural language communication*. Springer, Berlin, 2006.
311. M. Havey. *Essential business process modeling*. O'Reilly, 2005.
312. D. C. Hay. *Data model pattern: Conventions of thought*. Dorset House, New York, 1995.
313. D. Heckmann and A. Krüger. A user modeling markup language (userml) for ubiquitous computing. In *User Modeling*, pages 393–397, 2003.
314. L. J. Heinrich, A. Heinzl, and F. Roithmayr. *Wirtschaftsinformatik-Lexikon (7. Aufl.)*. Oldenbourg, 2004.
315. M. G. Helander. *A guide to human factors and ergonomics (2. ed.)*. CRC Press, 2006.
316. J. Henno. Emergence of language: hidden states and local environments. In *Information Modelling and Knowledge Bases XIX*, pages 170–180. IOS Press, 2008.
317. A. Heuer. *Zur Rolle generischer Operationen in objektorientierten Datenbanken*. infix, Sankt Augustin, 1994.
318. A. R. Hevner. The three cycle view of design science. *Scandinavian J. Inf. Systems*, 19(2), 2007.
319. J.-M. Hick and J.-L. Hainaut. Strategy for database application evolution: The DB-MAIN approach. In *ER 2003, Proceedings*, volume 2813 of *Lecture Notes in Computer Science*, pages 291–306. Springer, 2003.
320. C. Hoare and J. He. *Unifying theories of programming*. Prentice-Hall, Upper Saddle River, 1997.
321. C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969.
322. S. Hoberman. *Data Modeling Made Simple: A Practical Guide for Business & Information Technology Professionals*. John Wiley & Sons, New York, 2005.
323. G. Hofstede. Cultural dimensions. <http://www.geert-hofstede.com>, 2003.
324. G. Hofstede. Geert hofstede resource pages. <http://www.geert-hofstede.com>, Retrieved December 4th, 2014.
325. G. Hofstede. Cultural dimensions - WWW. <http://www.geert-hofstede.com>, Retrieved November 20th, 2013.
326. G. Hofstede and G.-J. Hofstede. *Cultures and Organizations: Software of the Mind: Intercultural Cooperation and Its Importance for Survival*. McGraw-Hill, New York, 2004.
327. G. Hofstede, G. J. Hofstede, and M. Minkow. *Cultures and Organizations: Software of the Mind: Intercultural Cooperation and Its Importance for Survival*. McGraw-Hill, New York, 2010.
328. U. Hohenstein. *Formale Semantik eines erweiterten Entity-Relationship-Modells*. Teubner, Stuttgart, 1993.
329. D. Homberger. *Sachwörterbuch der Sprachwissenschaft*. Reclam Universalbibliothek, 2003.
330. K. Höök, J. Karlsgren, A. Wærn, N. Dahlbäck, C. Gustaf Jansson, K. Karlsgren, and B. Lemaire. A glass box approach to adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 6(2):157–184, 1996.

331. I. Horrocks. *Constructing the user interface with statecharts*. Addison-Wesley, Harlow, 1999.
332. J. F. Horty. Deontic logic as founded on nonmonotonic logic. *Ann. Math. Artif. Intell.*, 9(1-2):69–91, 1993.
333. G.-J. Houben, P. Barna, F. Frasincar, and R. Vdovjak. HERA: Development of semantic web information systems. In *Third International Conference on Web Engineering – ICWE 2003*, volume 2722 of *LNCS*, pages 529–538. Springer-Verlag, 2003.
334. G.-J. Houben, K. van der Shuijs, P. Barna, J. Broekstra, S. Casteleyn, Z. Fiala, and F. Frasincar. HERA. In G. Rossi, O. Pastor, D. Schwabe, and L. Olsina, editors, *Web Engineering: Modelling and Implementing Web Applications*, pages 263–301. Springer-Verlag London Limited, 2008.
335. J. Hromkovič. *Communication Protocol Models*, pages 7–149. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.
336. R. Hu and P. Pu. A study on user perception of personality-based recommender systems. In *UMAP*, volume 6075 of *Lecture Notes in Computer Science*, pages 291–302, 2010.
337. M. E. C. Hull, K. Jackson, and J. Dick. *Requirements Engineering*. Springer, 2005.
338. W. Humphrey. *Managing the Software Process*. Addison-Wesley, 1989.
339. IBM (International Business Machines Corporation). Systems application architecture common user access / advanced interface design guide, 1991. No. SC34-4290.
340. IFML. Interaction flow modeling language. <http://www.omg.org/spec/IFML/>, Accessed Dec. 09, 2016. Accessed Oct 6, 2016.
341. N. Ind and C. M. Riondino. Branding on the web: A real revolution? *Journal of Brand Management*, 9(1):8–19, 2001.
342. K. Indermark and H. Klaeren. Efficient implementation of structural recursion. In *Proc. FCT'87*, volume 278 of *Lecture Notes in Computer Science*, pages 204–213. Springer, 1987.
343. Integranova. The product integranova m.e.s. <http://www.integranova.com/>, 2016. Accessed Dec. 09, 2016.
344. E. T. Irons. A syntax directed compiler for ALGOL 60 (reprint). *Commun. ACM*, 26(1):14–16, 1983.
345. T. Isakowitz, E. A. Stohr, and P. Balasubramanian. RMM: A methodology for structured hypermedia design. *Commun. ACM*, 38(8):34–44, 1995.
346. ISO. TR 15504-1: Information technology & software process assessment (part 1: Concepts and introductory guide). ISO/IEC, 1998-08-15, 1998.
347. ISO/IEC. Information technology - process assesment. parts 1-5. IS 15504, 2003-2006.
348. ISO/IEC. Information technology - process assessment - part 5: An exemplar process assessment model. FCD 15504-5:2004, 2004. not publicly available.
349. ISO/IEC. 25020 (Software and system engineering ? software product quality requirements and evaluation (square) ? measurement reference model and guide). ISO/IEC JTC1/SC7 N3280, 2005.
350. ISO/IEC JTC1/SC34. Topic Maps - Data Model. <http://www.isotopicmaps.org/sam/sam-model/>, June 2012.
351. T. Issa and P. Isaias. *Models and Methodologies*, pages 87–143. Springer London, London, 2015.

352. J. Itten. *Kunst der Farbe*. O. Maier, 1961.
353. H. Jaakkola, A. Heimbürg, and P. Linna. Knowledge oriented software engineering process in multi-cultural context. *Software Quality Journal*, 18(2):299–319, 2010.
354. H. Jaakkola, J. Henno, B. Thalheim, and J. Mäkelä. Collaboration, distribution and culture – challenges for communication. In *MiPRO*, pages 657–664. IEEE, 2015.
355. H. Jaakkola, T. Mäkinen, B. Thalheim, and T. Varkoi. Evolving the database co-design framework by SPICE. In *Information Modelling and Knowledge Bases Vol. XVII, Series Frontiers in Artificial Intelligence, volume 136*, pages 268–279. IOS Press, May 2006.
356. H. Jaakkola and B. Thalheim. Visual SQL – high-quality er-based query treatment. In *IWCMQ'2003*, volume 2814 of *LNCS*, pages 129–139. Springer, 2003.
357. H. Jaakkola and B. Thalheim. Software quality and life cycles. In *ADBIS 2005*, pages 208–220, Tallinn, September 2005. Springer.
358. H. Jaakkola and B. Thalheim. A framework for high quality software design and development: A systematic approach. *IET Software*, pages 105–118, April 2010.
359. H. Jaakkola and B. Thalheim. Architecture-driven modelling methodologies. In *Information Modelling and Knowledge Bases*, volume XXII, pages 97–116. IOS Press, 2011.
360. H. Jaakkola and B. Thalheim. Multicultural adaptive systems. In *Information Modelling and Knowledge Bases*, volume XXVI of *Frontiers in Artificial Intelligence and Applications*, 272, pages 172–191. IOS Press, 2014.
361. H. Jaakkola and B. Thalheim. Culture-aware web information system development. In *EJC'2015*, pages 121–138, Maribor, 2015.
362. H. Jaakkola and B. Thalheim. Culture-adaptable web information systems. In *Information Modelling and Knowledge Bases XXVII, Frontiers in Artificial Intelligence and Applications*, 280, pages 77–94. IOS Press, 2016.
363. H. Jaakkola and B. Thalheim. Recognising the culture context in information search. In *EJC'2016*, pages 167–185, Tampere, 2016.
364. H. Jaakkola and B. Thalheim. Supporting culture-aware information search. In *Information Modelling and Knowledge Bases XXVIII, Frontiers in Artificial Intelligence and Applications*, 280, pages 163–183. IOS Press, 2017.
365. H. Jaakkola, B. Thalheim, Y. Kidawara, K. Zettsu, Y. Chen, and A. Heimbürg. Information modelling and global risk management systems. In H. Jaakkola and Y. Kiyoki, editors, *Information Modeling and Knowledge Bases XX*, pages 429–446. IOS Press, 2009.
366. M. Jackson. *Problem Frames: Analysing and structuring software development problems*. Pearson Education, Harlowe, 2007.
367. J. Jacobsen. *Website-Konzeption*. Addison-Wesley, München, 2007.
368. K. Jannaschk, H. Jaakkola, and B. Thalheim. A framework for systematic change management. In *EuroSPI*, volume 425 of *Communications in Computer and Information Science*, pages 60–72. Springer, 2014.
369. K. Jannaschk, H. Jaakkola, and B. Thalheim. Technologies for database change management. In *ADBIS Vol. II*, volume 312 of *Advances in Intelligent Systems and Computing*, pages 271–286. Springer, 2014.
370. K. Jannaschk, C. A. Rathje, B. Thalheim, and F. Förster. A generic database schema for cidoc-crm data management. In *ADBIS (2)*, volume 789 of *CEUR Workshop Proceedings*, pages 127–136. CEUR-WS.org, 2011.

371. K. P. Jantke, S. Lange, G. Grieser, P. A. Grigoriev, B. Thalheim, and B. Tschiedel. Learning by doing and learning when doing: Dovetailing e-learning and decision support with a data mining tutor. In *Proceedings of the 6th International Conference on Enterprise Information Systems*, pages 238–241, Porto, Portugal, 2004.
372. K. P. Jantke, B. Schmidt, and R. Schnappauf. Next generation learner modeling by theory of mind model induction. In *Proc. CSEDU 2016*, pages 499–506. SciTePress, 2016.
373. M. Jantzen. *Basics of Term Rewriting*, pages 269–337. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.
374. C. Jennings. Speaking your mind: Using elements of narrative storytelling in elearning. *eLearn Magazine*, 2014(4):3, 2014.
375. J.H.Martin. *A Computational Model of Metaphor Interpretation*. Academic Press, Boston /San Diego/New York, 1990.
376. P. Jipsen. *Concurrent Kleene Algebra with Tests*, pages 37–48. Springer International Publishing, Cham, 2014.
377. P. Johannesson and E. Perjons. *An Introduction to Design Science*. Springer, Berlin, 2014.
378. M. Johnson. Seizing the white space: Business model innovation for growth and renewal. *Harward Business Review process*, 2010.
379. S. Johnson. *Interface culture*. Harper, San Francisco, 1997.
380. W. Jones. *Keeping found things found*. Morgan Kaufmann, 2008.
381. E. Jüngel. *Metapher*, chapter Metaphorische Wahrheit. Erwägungen zur theologischen Relevanz der Metapher als Beitrag zur Hermeneutik einer narrativen Theologie. De Gruyter, München, 1964.
382. I. Jureta. *The Design of Requirements Modelling Languages - How to Make Formalisms for Problem Solving in Requirements Engineering*. Springer, 2015.
383. Y. Kabutoya, T. Iwata, and K. Fujimura. Modeling multiple users' purchase over a single account for collaborative filtering. In *WISE*, pages 328–341, 2010.
384. B. K. Kahn. Requirement specification techniques. In S. B. Yao, editor, *Principles of database design, Volume I: Logical organizations*, pages 1–66. Prentice-Hall, Englewood Cliffs, 1985.
385. L. K. Kan, X. Peng, and I. King. A user profile-based approach for personal information access: shaping your information portfolio. In *WWW*, pages 921–922, 2006.
386. P. C. Kanellakis. Elements of relational database theory. In J. V. Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. B, Formal Models and Semantics*, pages 1074–1156. Elsevier, Amsterdam, 1990.
387. H. Kangassalo. Foundation of conceptual modeling: A theory construction view. *Information Modeling and Knowledge Bases*, pages 19–35, 1990.
388. H. Kangassalo. Approaches to the active conceptual modelling of learning. In *Active Conceptual Modeling of Learning*, LNCS 4512, pages 168–193, Berlin, 2007. Springer.
389. P. K. Kannan, A.-M. Chang, and A. B. Whinston. Electronic communities in e-business: Their role and issues. *Information Systems Frontiers*, 1(4):415–426, 2000.
390. G. Kappel, B. Pröll, S. Reich, and W. Retschitzegger. *Web Engineering*. John Wiley & Sons, Heidelberg, 2006.
391. V. Kaptelinin, K. Kuutti, and L. J. Bannon. Activity theory: Basic concepts and applications. In *EWHCI*, pages 189–201, 1995.

392. D. Karagiannis, H. Mayr, and J. Mylopoulos, editors. *Domain-specific conceptual modeling: Concepts, methods and tools*. Springer, Cham, 2016.
393. E. Karttunen, H. Jaakkola, and P. Linna. Effects of cross-cultural factors in global services production. In *PICMET 2011*, pages 2552–2561, Portland, 2011.
394. R. Kaschek, K.-D. Schewe, B. Thalheim, T. Kuss, and B. Tschedel. Learner typing for electronic learning systems. In Kinshuk, C.-K. Looi, E. Sutinen, D. G. Sampson, I. Aedo, L. Uden, and E. Kähkönen, editors, *Proceedings of the IEEE International Conference on Advanced Learning Technologies (ICALT 2004)*, pages 375–379. IEEE Computer Society, 2004.
395. R. Kaschek, K.-D. Schewe, B. Thalheim, and L. Zhang. Integrating context in conceptual modelling for web information systems. In *Web Services, E-Business, and the Semantic Web*, volume 3095 of *LNCS*, pages 77–88. Springer-Verlag, 2004.
396. R. Kaschek, K.-D. Schewe, C. Wallace, and C. Matthews. Story boarding for web information systems. In D. Taniar and W. Rahayu, editors, *Web Information Systems*, pages 1–33. IDEA Group, 2004.
397. R. H. Kaschek, C. Matthews, K.-D. Schewe, and C. Wallace. Information systems design: through adaptivity to ubiquity. *Inf. Syst. E-Business Management*, 4(2):137–158, 2006.
398. R. H. Kaschek, C. Matthews, C. Wallace, and K.-D. Schewe. Increasing automation in lending. In G. Kotsis, S. Bressan, and I. K. Ibrahim, editors, *iiWAS'2003 - The Fifth International Conference on Information Integration-and Web-based Applications Services*, volume 170 of *books@ocg.at*. Austrian Computer Society, 2003.
399. R. H. Kaschek, K.-D. Schewe, and B. Thalheim. Context modeling for web information systems. In J. Eder and T. Welzer, editors, *The 15th Conference on Advanced Information Systems Engineering (CAiSE 2003)*, volume 74 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2003.
400. R. H. Kaschek, K.-D. Schewe, B. Thalheim, T. Kuss, and B. Tschedel. Learner typing for electronic learning systems. In *Proc. ICALT 2004*, pages 1220–1231. IEEE Computer Society, 2004.
401. R. H. Kaschek, K.-D. Schewe, B. Thalheim, and L. Zhang. Integrating context in modelling for web information systems. In C. Bussler, D. Fensel, M. E. Orlowska, and J. Yang, editors, *Web Services, E-Business, and the Semantic Web, Second International Workshop (WES 2003)*, volume 3095 of *Lecture Notes in Computer Science*, pages 77–88. Springer, 2003.
402. V. Kashyap, C. Bussler, and M. Moran. *The Semantic Web - Semantics for Data and Services on the Web*. Springer, 2008.
403. T. Kato. User modeling through unconscious interaction with smart shop. In *HCI (6)*, volume 6766 of *Lecture Notes in Computer Science*, pages 61–68, 2011.
404. R. Kauppi. Einführung in die Theorie der Begriffssysteme. *Acta Universitatis Tamperensis, Ser. A, Vol. 15, Tampereen yliopisto, Tampere*, 1967.
405. Z. Kedad and E. Métais. Dealing with semantic heterogeneity during data integration. In J. Akoka, M. Bouzeghoub, I. Comyn-Wattiau, and E. Métais, editors, *Conceptual Modeling – ER'99*, volume 1728 of *LNCS*, pages 325–339. Springer-Verlag, 1999.
406. M. Keidl and A. Kemper. *A Framework for Context-Aware Adaptable Web Services*, pages 826–829. Springer, Berlin, 2004.

407. F. Kensing and J. Blomberg. Participatory design: Issues and concerns. *Computer Supported Cooperative Work*, 7(3/4):167–185, 1998.
408. I. O. Kerner and H. Al-Sheikh-Khalil. Algorithmische sprache ALGOL 60+8. *Elektronische Informationsverarbeitung und Kybernetik*, 10(2/3):71–107, 1974.
409. Y. Kidawara, K. Zettsu, Y. Kiyoki, K. Jannaschk, B. Thalheim, P. Linna, H. Jaakkola, and M. Duzi. Knowledge modeling, management and utilization towards next generation web. In *Information Modelling and Knowledge Bases XXI*, volume 206, pages 387–402. IOS Press, 2010.
410. K.-S. Kim and Y. Kiyoki. An object-field perspective data model for moving geographic phenomena. In *DASFAA Workshops*, volume 6193 of *Lecture Notes in Computer Science*, pages 410–421. Springer, 2010.
411. W. King. A research agenda for the relationships between culture and knowledge management. *Knowledge and Process Management*, 14(3):226 – 236, 2007.
412. M. Kirchberg, F. Riaz-ud-Din, and K.-D. Schewe. A query language for rational tree structures. In *Proc. UNISCON 2008*, volume 5 of *Lecture Notes in Business Information Processing*, pages 457–468. Springer, 2008.
413. M. Kirchberg, F. Riaz-ud-Din, K.-D. Schewe, and A. Tretiakov. Using reflection for querying XML documents. In *Proc. ADC 2006*, volume 49 of *CRPIT*, pages 119–128. Australian Computer Society, 2006.
414. M. Kirchberg, K.-D. Schewe, B. Thalheim, and R. Wang. A three-level architecture for distributed web information systems. In N. Koch, P. Fraternali, and M. Wirsing, editors, *Web Engineering - 4th International Conference (ICWE 2004)*, volume 3140 of *Lecture Notes in Computer Science*, pages 137–141. Springer, 2004.
415. M. Kirchberg, K.-D. Schewe, and A. Tretiakov. A multi-level architecture for distributed object bases. In *Proceedings of the 5th International Conference on Enterprise Information Systems (ICEIS)*, volume 1, pages 63–70. ICEIS Press, 2003.
416. M. Kirchberg, K.-D. Schewe, and A. Tretiakov. Using XML to support media types. In *Proc. ISTA'2005*, volume 63 of *LNI*, pages 101–113. GI, 2005.
417. M. Kirchberg, K.-D. Schewe, A. Tretiakov, and R. Wang. A multi-level architecture for distributed object bases. *Data Knowl. Eng.*, 60(1):150–184, 2007.
418. M. Kirchberg, O. Sørensen, and B. Thalheim. A BPMN case study: Paper review and submission system. In *GI Jahrestagung*, volume 154 of *LNI*, pages 4067–4081. GI, 2009.
419. E. Kittay. *Metaphor: Its Cognitive Force and its Linguistic Structure*. Clarendon Press, Oxford, 1987.
420. Y. Kiyoki and B. Thalheim. Analysis-driven data collection, integration and preparation for visualisation. In *Information Modelling and Knowledge Bases*, volume XXIV, pages 142–160. IOS Press, 2013.
421. S. C. Kleene. Representation of events in nerve sets and finite automata. In Shannon and McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press, 1956.
422. C. Kleiner and U. Lipeck. Automatic generation of XML DTDs from conceptual database schemas. In *GI Jahrestagung (1)*, pages 396–405, 2001.
423. B. H. Kleint. *Bildlehre*. Schwabe & Co., Basel, 1929.
424. M. Klettke. *Modellierung, Bewertung und Evolution von XML-Dokumentkollektionen*. Advanced PhD (Habilitation Thesis), Rostock University, Faculty for Computer Science and Electronics, 2007.

425. M. Klettke and H. Meyer. *XML & Datenbanken - Konzepte, Sprachen und Systeme*. dpunkt.verlag, Heidelberg, 2003.
426. M. Klettke and B. Thalheim. Evolution and migration of information systems. In *The Handbook of Conceptual Modeling: Its Usage and Its Challenges*, chapter 12, pages 381–420. Springer, Berlin, 2011.
427. R. Kluwe. *Allgemeine Psychologie*, chapter Gedächtnis und Wissen, pages 115–187. Bern, 1990.
428. R. Knauf and K. P. Jantke. Towards an evaluation of (e-)learning systems. In *Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference, Clearwater Beach, Florida, USA*, pages 226–231. AAAI Press, 2005.
429. R. Knauf and K. K. P. Jantke. Didactic design through storyboarding: Standard concepts for standard tools. In *Proc. of the 4th International Symposium on Information and Communication Technologies, Workshop on Dissemination of e-Learning Technologies and Applications, Cape Town, South Africa, 2005*, pages 20–25, 2005.
430. M. Knott. *Table and Collection Views*, pages 213–276. Apress, Berkeley, CA, 2014.
431. D. E. Knuth and P. B. Bendix. Simple word problems in universal algebras. In *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, Oxford, UK, 1970.
432. A. Kobsa. User modeling and user-adapted interaction. *User Modeling and User-Adapted Interaction*, 15(1-2):185–190, 2005.
433. A. Kobsa and W. Pohl. Workshop an adaptivity and user modeling in interactive software systems. *User Model. User-Adapt. Interact.*, 3(4):359–367, 1994.
434. K. Koffa. *Principles of Gestalt Psychology*. Routledge and Kegan Paul, London, 1935.
435. J. Koh and A. Chen. Integration of heterogeneous object schemas. In R. Elmasri, V. Kouramajian, and B. Thalheim, editors, *Entity-Relationship Approach - ER'93*, volume 823 of *LNCS*, pages 297–314. Springer-Verlag, 1994.
436. W. Köhler. *Gestalt Psychology*. H. Liveright, New York, 1929.
437. M. Köhnke, T. Ignatova, M. Weicht, and I. Bruder. Identifying semantic constructs in web documents to improve web site accessibility. In *WISE Workshops*, volume 5176 of *Lecture Notes in Computer Science*, pages 92–101. Springer, 2008.
438. A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1):1–7, 1965.
439. H. König. *Protocol Engineering: Prinzip, Beschreibung und Entwicklung von Kommunikationsprotokollen*. Teubner, Stuttgart, 2003.
440. R. Koper and C. Tattersall, editors. *Learning design: A handbook of modelling and delivering networked education and training*. Springer, Heidelberg, 2005.
441. K. Korta and J. Perry. *Critical Pragmatics*. Cambridge University Press, Cambridge, 2011.
442. A. Kosmaczewski. *Mobile JavaScript Application Development - Bringing Web Programming to Mobile Devices*. O'Reilly, 2012.
443. F. Kossak, C. Illibauer, V. Geist, J. Kubovy, C. Natschläger, T. Ziebermayr, T. Kopetzky, B. Freudenthaler, and K.-D. Schewe. *A Rigorous Semantics for BPMN 2.0 Process Diagrams*. Springer, 2014.

444. F. Kossak, C. Illibauer, V. Geist, C. Natschläger, T. Ziebermayr, B. Freudenthaler, T. Kopetzky, and K.-D. Schewe. *Hagenberg Business Process Modelling Method*. Springer, 2016.
445. P. Kotler. B2B branding dimensions. In P. Kotler and W. Pfoertsch, editors, *B2B Branding Management*, pages 65–156. Springer, Berlin, 2006.
446. Y. Kou, D. Shen, T. Nie, and G. Yu. Potential role based entity matching for dataspaces search. In *WISE*, pages 496–509, 2010.
447. Z. Kövecses. *Metaphor: A Practical Introduction*. Oxford University Press, Oxford, UK, 2002.
448. J. A. Kowal. *Behavior models - specifying user's expectations*. Prentice Hall, 1992.
449. D. Kozen. On Kleene algebra and closed semirings. In *Mathematical Fundamentals of Computer Science*, pages 26–47, 1990.
450. D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. In *Logic in Computer Science*, pages 214–225, 1991.
451. D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information & Computation*, 110(2):366–390, 1994.
452. D. Kozen. Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems*, 19(3):427–443, 1997.
453. D. Kozen. On Hoare logic and Kleene algebra with tests. In *Logic in Computer Science*, pages 167–172, 1999.
454. D. Kozen. On the complexity of reasoning in Kleene algebra. *Information and Computation*, 179(2):152–162, 2002.
455. D. Kozen. *On the Representation of Kleene Algebras with Tests*, pages 73–83. Springer, Berlin, 2006.
456. D. Kozen and F. Smith. Kleene algebra with tests: Completeness and decidability. In *Computer Science Logic*, pages 244–259, 1996.
457. F. Kramer and B. Thalheim. Component-based development of a metadata data-dictionary. In *BIS*, volume 176 of *Lecture Notes in Business Information Processing*, pages 110–121. Springer, 2014.
458. F. Kramer and B. Thalheim. A metadata system for quality management. In *Information Modelling and Knowledge Bases*, volume XXVI of *Frontiers in Artificial Intelligence and Applications*, 272, pages 224–242. IOS Press, 2014.
459. F. Kramer and B. Thalheim. Data provenance management based on metadata. In *EJC'2016*, pages 199–218, Tampere, 2016.
460. F. Kramer and B. Thalheim. Holistic conceptual and logical database structure modelling with adoxx. In *Domain-specific conceptual model*, pages 269–290, Cham, 2016. Springer.
461. T. Kreifelts, E. Hinrichs, and G. Woetzel. Bscw-flow: Workflow in web-based shared workspaces. In *Cross-Organisational Workflow Management and Coordination*, 1999.
462. J. Kretschmer, C. Schranz, A. Riedlinger, D. Schädler, and K. Möller. *Wissenschaft und Kunst der Modellierung: Modelle, Modellieren, Modellierung*, chapter Hierarchische Modellsysteme zur Optimierung der Beatmungstherapie, pages 369–389. De Gryuter, Boston, 2015.
463. C. Kroiss, N. Koch, and A. Knapp. UWE4JSF: A model-driven generation approach for web applications. In *Proc. ICWE 2009*, volume 5648 of *Lecture Notes in Computer Science*, pages 493–496. Springer, 2009.

464. S. J. Krolkoski, F. Schirrmeister, B. Salefski, J. Rowson, and G. Martin. Methodology and technology for virtual component driven hardware/software co-design on the system-level. In *Proc. ISCAS*, pages 456–459. IEEE, 1999.
465. P. Kruchten. *The Rational Unified Process - An Introduction*. Addison-Wesley, 1998.
466. P. Kruchten. Use-case storyboards in the rational unified process. In *Proc. ECOOP'99 Workshops*, volume 1743 of *Lecture Notes in Computer Science*, pages 249–250. Springer, 1999.
467. J. D. Krüger. *Web Content managen. Professioneller Einsatz von Content-Management-Systemen*. Markt + Technik, München, 2002.
468. H. Kubicek, U. Horst, V. Redder, U. Schmid, I. Schumann, W. Taube, and H. Wagner. www.stadtinfo.de. Hüthig, 1997.
469. B. Kuechler and V. Vaishnavi. On theory development in design science research: Anatomy of a research project. *European Journal of Information Systems*, 17(5):489–504, 2008.
470. T. Kühme, H. Dieterich, U. Malinowski, and M. Schneider-Hufschmidt. Approaches to adaptivity in user interface technology: Survey and taxonomy. In *Engineering for Human-Computer Interaction, Proc. IFIP TC2/WG2.7*, volume A-18 of *IFIP Transactions*, pages 225–252. North-Holland, 1992.
471. D. Kumbier and F. von Thun. *Interkulturelle Kommunikation: Methoden, Modelle, Beispiele*. rororo, 2006.
472. J. Kunze. Generating verb fields. In *Proc. KONVENTS*, Informatik Aktuell, pages 268–277. Springer, 1992. in German.
473. G. Kurz. *Metapher, Theorie und Unterricht*, chapter Die Rhetorik der Metapher. Schwan, Düsseldorf, 1976.
474. E. Küthe and M. Thun. *Marketing mit Bildern: Management mit Trend-Tableaus, Mood-Charts, Storyboards, Fotomontagen und Collagen*. DuMont, Köln, 1995.
475. A. Labrinidis and Y. Sismanis. *View Maintenance*, pages 3326–3328. Springer US, Boston, MA, 2009.
476. G. Lakoff. *Women, Fire and Dangerous Things*. University of Chicago Press, Chicago, U.S.A., 1987.
477. G. Lakoff and M. Johnson. *Metaphors We Live By*. University of Chicago Press, Chicago, U.S.A., 1980.
478. N. Lammari, J.-S. Bucumi, J. Akoka, and I. Comyn-Wattiau. A conceptual meta-model for secured information systems. In *Proc. 7th International Workshop Software Engineering for Secure Systems, SESS 2011*, pages 22–28. ACM, 2011.
479. L. Landwehr, B. Thalheim, and A. Vitzthum. Navigieren in Wissensstrukturen - Benutzerprofile und neue Retrievalstrukturen. In *Tagung Kunst- und Medientechnologie, Karlsruhe*, pages 8–9, 2007.
480. H. Langmaack. On procedures as open subroutines. I. *Acta Inf.*, 2:311–333, 1973.
481. H. Langmaack. On procedures as open subroutines. II. *Acta Inf.*, 3:227–241, 1974.
482. A. N. Langville and C. D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, 2012.
483. M. W. Lansdale and T. C. Ormerod. *Understanding interfaces, a handbook of human-computer dialogue*. Academic Press, London, 1994.

484. Laofutze. Notes on intercultural communication. Available in <https://laofutze.wordpress.com/2009/07/28/e-t-hall-high-low-context/>, 2016. Retrieved October 19th, 2016.
485. G. Larsen and J. Conallen. Engineering web-based systems with UML assets. *Ann. Software Eng.*, 13(1-4):203–230, 2002.
486. J. Larson and C. L. Larson. Evaluation of four languages for specifying conceptual database designs. *High-Performance Web Databases 2001*, pages 171–194, 2001.
487. J. Larson, S. B. Navathe, and R. Elmasri. A theory of attribute equivalence in databases with application to schema integration. *IEEE Transactions on Software Engineering*, 15(4):449–463, 1989.
488. P.-Å. Larson. *Maintenance of Materialized Views with Outer-Joins*, pages 1670–1674. Springer US, Boston, MA, 2009.
489. B. Laurel. *The art of human computer interface design*. Addison-Wesley, 1990.
490. Y.-N. Law, H. Wang, and C. Zaniolo. Relational languages and data models for continuous queries on sequences and data streams. *ACM Trans. Database Syst.*, 36(2):8, 2011.
491. D. Lawrence and S. Tavakol. *Balanced website design - optimising aesthetics, usability and purpose*. Springer, 2007.
492. R. Lawrence, G. Almasi, V. Kotlyar, M. Viveros, and S. Duri. Personalization of supermarket product recommendations. *Data Mining and Knowledge Discovery*, 5(1):11–32, 2001.
493. T. Lehmann and K.-D. Schewe. A pragmatic method for the integration of higher-order Entity-Relationship schemata. In A. H. F. Laender, S. W. Liddle, and V. C. Storey, editors, *Conceptual Modeling - ER 2000*, volume 1920 of *LNCS*, pages 37–51. Springer-Verlag, 2000.
494. W. Lehner. *Query Processing in Data Warehouses*, pages 2297–2301. Springer US, Boston, MA, 2009.
495. H.-J. Lenz and B. Thalheim. OLAP databases and aggregation functions. In *Proc. SSDBM 2001*, pages 91–100. IEEE, 2001.
496. H.-J. Lenz and B. Thalheim. OLTP-OLAP schemes for sound applications. In *TEAA 2005*, volume LNCS 3888, pages 99–113, Trondheim, 2005. Springer.
497. H.-J. Lenz and B. Thalheim. A formal framework of aggregation for the OLAP-OLTP model. *J. UCS*, 15(1):273–303, 2009.
498. A. Leontiev. *Activity, consciousness, and personality*. Prentice Hall, Englewood Cliffs, 1978.
499. M. Lepasaar, T. Mäkinen, and T. Varkoi. Structural comparison of SPICE and continuous CMMI. In A. Dorling, F. Fabbrini, M. Fusani, and T. Rout, editors, *SPICE 2002*, Venice, Italy, 2002.
500. M. Levene and G. Loizou. *A guided tour of relational databases and beyond*. Springer, Berlin, 1999.
501. J. Lewerenz, K.-D. Schewe, and B. Thalheim. Modelling data warehouses and OLAP applications using dialogue objects. In J. Akoka, M. Bouzeghoub, I. Comyn-Wattiau, and E. Métais, editors, *Conceptual Modeling – ER'99*, volume 1728 of *LNCS*, pages 354–368. Springer-Verlag, 1999.
502. R. Lewis. *When Cultures Collide. Managing Successfully Across Cultures*. Nicholas Brealey, London, 3rd edition, 2011.
503. R. Lewis. Richard Lewis resource pages - Cross-culture. <http://www.crossculture.com/services/cross-culture/> & <http://www.cultureactive.com>, 2013. Retrieved November 20th, 2013.

504. R. Lewis. Richard Lewis resource pages - Cross-culture. <http://www.crossculture.com/services/cross-culture/> & <http://www.cultureactive.com>, 2014. Retrieved December 4th, 2014.
505. R. Lewis. Richard Lewis resource pages - cross-culture. <http://www.cultureactive.com>, 2016. Retrieved February 1st, 2016.
506. S. W. Liddle, D. W. Embley, and S. N. Woodfield. Cardinality constraints in semantic data models. *DKE*, 11:235–270, 1993.
507. A. Light. Designing to persuade: the use of emotion in networked media. *Interacting with Computers*, 16(4):729–738, 2004.
508. S. Lightstone, T. Teorey, and T. Nadeau. *Physical database design*. Morgan Kaufmann, 2007.
509. E. Lindsey. *Three Dimensional Analysis - Data Profiling Techniques*. Data Profiling LLC, 2008.
510. LinkedIn. Discussion about modelling tools. http://www.linkedin.com/groupItem?view=&srchtype=discussedNews&gid=988447&item=29745443&type=member&trk=eml-anet_dig-b_pd-ttl-cn, 2016, accessed Sept. 30, 2016.
511. H. Lobin. *Informationsmodellierung in XML und SGML*. Springer-Verlag, 2000.
512. P. Lockemann. Information system architectures: From art to science. In *Proc. BTW'2003, Springer, Berlin*, pages 1–27, 2003.
513. P. C. Lockemann and G. Moerkotte. On the notion of concept. In *ER*, pages 349–370. ER Institute, 1991.
514. P. C. Lockemann, G. Moerkotte, A. Neufeld, K. Radermacher, and N. Runge. Database design with user-definable modelling concepts. *Data Knowl. Eng.*, 10:229–257, 1993.
515. G. Lolli. *Recursion theory and computational complexity*. Liguori, 1981.
516. Lom. ieee learning technology standards committee's (ltsc) lom working group. learning object meta-data (lom). <http://www.imsproject.org>, 2000.
517. P. Loos, R. Riedl, G. R. Müller-Putz, J. vom Brocke, F. D. Davis, R. D. Bunker, and P.-M. Léger. Neurois: Neuroscientific approaches in the investigation and development of information systems. *Business & Information Systems Engineering*, 2(6):395–401, 2010.
518. D. Lowe, B. Henderson-Sellers, and A. Gu. Web extensions to UML: Using the MVC triad. In S. Spaccapietra, S. T. March, and Y. Kambayashi, editors, *Conceptual Modeling – ER 2002*, volume 2503 of *LNCS*, pages 105–119. Springer-Verlag, 2002.
519. LTSC. Learning technology standards committee website. <http://ltsc.ieee.org/>, 2000.
520. E. R. Luna, I. Garrigós, J.-N. Mazón, J. Trujillo, and G. Rossi. An i*-based approach for modeling and testing web requirements. *J. Web Eng.*, 9(4):302–326, 2010.
521. E. R. Luna, G. Rossi, and I. Garrigós. Webspec: a visual language for specifying interaction and navigation requirements in web applications. *Requir. Eng.*, 16(4):297–321, 2011.
522. K. Luyten, K. Coninx, M. Abrams, and H. Inc. Integrating uiml, task and dialogs with layout patterns for multi-device user interface design. In *The 11th International Conference on Human-Computer Interaction, Las Vegas*, pages 22–27, 2005.

523. H. Ma, R. Noack, K.-D. Schewe, and B. Thalheim. Using meta-structures in database design. *Informatica*, 34:387–403, 2010.
524. H. Ma, R. Noack, K.-D. Schewe, B. Thalheim, and Q. Wang. Complete conceptual schema algebras. *Fundamenta Informaticae*, 123:1–26, 2013.
525. H. Ma and K.-D. Schewe. Fragmentation of XML documents. In *Proceedings XVIII Simpósio Brasileiro de Bancos de Dados (SBBD 2003)*, pages 200–214, Manaus, Brazil, 2003.
526. H. Ma, K.-D. Schewe, and B. Thalheim. Integration and cooperation of media types. In R. H. Kaschek, H. C. Mayr, and S. W. Liddle, editors, *Information Systems Technology and its Applications, 4th International Conference ISTA'2005*, volume 63 of *LNI*, pages 139–153. GI, 2005.
527. H. Ma, K.-D. Schewe, and B. Thalheim. Context analysis: Toward pragmatics of web information systems design. In A. Hinze and M. Kirchberg, editors, *Conceptual Modelling 2008, Fifth Asia-Pacific Conference on Conceptual Modelling (APCCM 2008)*, volume 79 of *CRPIT*, pages 69–78. Australian Computer Society, 2008.
528. H. Ma, K.-D. Schewe, and B. Thalheim. Modelling and maintenance of very large database schemata using meta-structures. In *UNISCON*, volume 20 of *Lecture Notes in Business Information Processing*, pages 17–28. Springer, 2009.
529. H. Ma, K.-D. Schewe, and B. Thalheim. Storyboarding - high-level engineering of web information systems. In G. Vossen, D. D. E. Long, and J. X. Yu, editors, *Web Information Systems Engineering (WISE 2009), 10th International Conference*, volume 5802 of *Lecture Notes in Computer Science*, pages 7–8. Springer, 2009.
530. H. Ma, K.-D. Schewe, and B. Thalheim. Web information systems design in the era of Web 2.0 and beyond. In G. Kotsis, D. Taniar, E. Pardede, and I. Khalil, editors, *iiWAS'2009 - The Eleventh International Conference on Information Integration and Web-based Applications and Services*, page 10. ACM, 2009.
531. H. Ma, K.-D. Schewe, B. Thalheim, and Q. Wang. Abstract state services. In I.-Y. Song et al., editors, *Advances in Conceptual Modeling - Challenges and Opportunities (ER 2008 Workshops)*, volume 5232 of *Lecture Notes in Computer Science*, pages 406–415. Springer, 2008.
532. H. Ma, K.-D. Schewe, B. Thalheim, and Q. Wang. A theory of data-intensive software services. *Service Oriented Computing and Applications*, 3(4):263–283, 2009.
533. H. Ma, K.-D. Schewe, B. Thalheim, and Q. Wang. Conceptual modelling of services. *Journal of Universal Computer Science*, 18(17):2361–2363, 2012.
534. H. Ma, K.-D. Schewe, B. Thalheim, and Q. Wang. A formal model for the interoperability of service clouds. *Service Oriented Computing and Applications*, 6(3):189–205, 2012.
535. H. Ma, K.-D. Schewe, B. Thalheim, and J. Zhao. View integration and cooperation in databases, data warehouses and web information systems. *Journal on Data Semantics IV*, 3730:213–249, 2005.
536. H. Ma, K.-D. Schewe, and Q. Wang. An abstract model for service provision, search and composition. In M. Kirchberg, P. C. K. Hung, B. Carminati, C.-H. Chi, R. Kanagasabai, E. D. Valle, K.-C. Lan, and L.-J. Chen, editors, *4th IEEE Asia-Pacific Services Computing Conference (IEEE APSCC 2009)*, pages 95–102. IEEE, 2009.
537. L. Maciaszek. *Requirements analysis and design*. Addison-Wesley, Harlow, Essex, 2001.

538. L. Maciaszek and B. Liong. *Practical software engineering*. Addison-Wesley, Harlow, Essex, 2005.
539. B. Magnini and C. Strapparava. User modelling for news web sites with word sense based techniques. *User Modeling and User-Adapted Interaction*, 14(2-3):239–257, 2004.
540. M. J. Maher. Complete axiomatizations of the algebras of finite, rational and infinite trees. In *Proc. (LICS '88)*, pages 348–357. IEEE Computer Society, 1988.
541. B. Mahr. Cargo. Zum Verhältnis von Bild und Modell. In I. Reichle, S. Siegel, and A. Spelten, editors, *Visuelle Modelle*, pages 17–40. Wilhelm Fink Verlag, München, 2008.
542. D. Maier. *The theory of relational databases*. Computer Science Press, Rockville, MD, 1983.
543. T. Mäkinen, T. Varkoi, and H. Jaakkola. Database implementation for a software process assessment model. In D. Kocaoglu and T. Anderson, editors, *PICMET'99, Portland International Conference on Management of Engineering and Technology*, Portland, Oregon, USA, July 25-29, 1999.
544. T. Mäkinen, T. Varkoi, and M. Lepasaar. A detailed process assessment method for software smes. In *EuroSPI 2000, Practical and Innovation Based Software Process Improvement to Prepare for the New Millenium*, pages 1–26, Copenhagen, Nov. 7-9 2000. Copenhagen Business School.
545. T. W. Malone and K. Crowston. The interdisciplinary study of coordination. *ACM Comput. Surv.*, 26(1):87–119, 1994.
546. A. I. Malzew. *Algebraic systems*. Nauka, Moscow, 1970.
547. C. Mancas. *Conceptual Data Modeling and Database Design: A Fully Algorithmic Approach*. Apple Academic Press, 2015.
548. J. Mandemakers. *Life Events*, pages 3560–3563. Springer Netherlands, Dordrecht, 2014.
549. H. Mannila and K.-J. Räihä. *The design of relational databases*. Addison-Wesley, Wokingham, England, 1992.
550. S. March and V. Storey. Design science in the information systems discipline: An introduction to the special issue on design science research. *MIS Quarterly*, 4:725–730, 2008.
551. D. Marco. *Building and managing the meta data repository*. Wiley Publ. Inc., 2000.
552. D. Marco and M. Jennings. *Universal meta data models*. Wiley Publ. Inc., 2004.
553. A. Marinos. *Generative web information systems*. PhD thesis, University of Surrey, Guildford, UK, 2011.
554. F. J. Martínez-López, editor. *Handbook of Strategic e-Business Management*. Springer, 2014.
555. H. Maurer, N. Scherbakov, Z. Halim, and Z. Razak. *From Databases to Hypermedia. With 26 CAI Lessons*. Springer, 1988.
556. H. C. Mayr and C. Kop. Conceptual predesign - bridging the gap between requirements and conceptual design. In *Proc. (ICRE '98)*, page 90. IEEE Computer Society, 1998.
557. H. C. Mayr, K.-D. Schewe, B. Thalheim, and T. Welzer. Integration of bargaining into e-business systems. *Informatica (Slovenia)*, 30(3):335–345, 2006.
558. J. Mazur. *Learning and behaviour*. Pearson, London, 2005.

559. P. McNamara. Deontic logic. In *Logic and the Modalities in the Twentieth Century*, volume 7 of *Handbook of the History of Logic*, pages 197–288. Elsevier, 2006.
560. G. Mecca, P. Merialdo, and P. Atzeni. ARANEUS in the era of XML. *IEEE Data Engineering Bulletin*, 1999.
561. P. B. Meggs. *Type and Image: The Language of graphic design*. John Wiley & Sons, 1992.
562. M. Meldrum and M. McDonald. *Branding*, pages 149–152. Macmillan Education UK, London, 1995.
563. S. Meliá and J. Gómez. The WebSA approach: Applying model driven engineering to web applications. *Journal of Web Engineering*, 5(2):121–149, 2006.
564. W. Menke. *Geophysical Data Analysis: Discrete Inverse Theory*, volume 45 of *International Geophysics*. Academic Press Inc., 1989.
565. J. Meseguer. *Rewriting Logic and Maude: Concepts and Applications*, pages 1–26. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.
566. B. Meyer. *Touch of Class Learning to Program Well with Objects and Contracts*. Springer, Heidelberg, 2009.
567. J.-J. C. Meyer. A different approach to deontic logic: deontic logic viewed as a variant of dynamic logic. *Notre Dame Journal of Formal Logic*, 29(1):109–136, 1988.
568. J.-J. C. Meyer, R. Wieringa, and F. Dignum. The role of deontic logic in the specification of information systems. In *Logics for Databases and Information Systems (the book grow out of the Dagstuhl Seminar 9529: Role of Logics in Information Systems, 1995)*, pages 71–115. Kluwer, 1998.
569. Z. Michalewicz and D. B. Fogel. *How to solve it: Modern heuristics*. Springer, Berlin, 1999.
570. G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63(2):81–97, 1956.
571. J. A. Mitchell. *Introduction: Conceptual Personae*, pages 1–9. Palgrave Macmillan US, New York, 2009.
572. MOFM2T. Mof model to text transformation language. <http://www.omg.org/spec/MOFM2T>, 2016. Accessed Dec. 09, 2016.
573. N. Mohd Yusoff and S. S. Salim. *A Review of Storyboard Tools, Concepts and Frameworks*, pages 73–82. Springer International Publishing, Cham, 2014.
574. P. Molina, S. Meliá, and O. Pastor. Just-UI : A user interface specification model. In *Computer-Aided Design of User Interfaces III (CADUI) 2002*, pages 63–74, Valenciennes, 2002. Kluwer.
575. A. Molnar. *A general partition data model and a contribution to the theory of functional dependencies*. PhD thesis, Eötvös Loránd University, Faculty of Informatics, Budapest, 2007.
576. R. Morales-Chaparro, J. C. Preciado, and F. Sánchez-Figueroa. Data-driven and user-driven multidimensional data visualization. In *ICWE Workshops*, volume 7059 of *Lecture Notes in Computer Science*, pages 159–166, 2011.
577. N. Moreno, J. R. Romero, and A. Vallecillo. On overview of model-driven web engineering and the MDA. In G. Rossi, O. Pastor, D. Schwabe, and L. Olsina, editors, *Web Engineering: Modelling and Implementing Web Applications*, pages 353–382. Springer, 2008.
578. K. Morfonios and Y. Ioannidis. *Cube Implementations*, pages 539–544. Springer US, Boston, MA, 2009.

579. T. Moritz. Skriptum - Grundlagen der Gestaltung visueller Medien. Media Design course, Computer Science Institute, Cottbus University of Technology, 1997.
580. T. Moritz. *Visuelle Gestaltungs raster interaktiver Informationssysteme als integrativer Bestandteil des immersiven Bildraumes*. PhD thesis, HFF Berlin-Babelsberg, 2006.
581. T. Moritz. *The screenography of interactive information systems – Lecture notes for papers (program: information and media technology)*. Brandenburg University of Technology, Cottbus, 2010–2013.
582. T. Moritz. *Inszenierung von Interaktivität: Screenography als integrativer Bestandteil systematischer Entwicklung und Gestaltung interaktiv nutzbarer Informations- und Kommunikationssysteme*. Lang, Berlin, 2016/17.
583. T. Moritz, R. Noack, K.-D. Schewe, and B. Thalheim. Intention-driven screenography. In H. C. Mayr and D. Karagiannis, editors, *Information Systems Technology and its Applications, 6th International Conference (ISTA 2007)*, volume 107 of *LNI*, pages 128–139. GI, 2007.
584. T. Moritz, R. Noack, K.-D. Schewe, and B. Thalheim. Principles of screenography. In J. Eder, S. L. Tomassen, A. L. Opdahl, and G. Sindre, editors, *CAiSE'07 Forum, Proceedings of the CAiSE'07 Forum at the 19th International Conference on Advanced Information Systems Engineering*, volume 247 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.
585. T. Moritz, K.-D. Schewe, and B. Thalheim. Strategic modeling of web information systems and its impact on visual design patterns. In F. Frasincar, G.-J. Houben, and R. Vdovjak, editors, *WISM'05*, pages 5–13, Sydney, 2005.
586. T. Moritz, K.-D. Schewe, and B. Thalheim. Strategic modelling of web information systems. *International Journal on Web Information Systems*, 1(4):223–240, 2005.
587. MPS. Mps overview. <https://www.jetbrains.com/mps/>, 2016. Accessed Dec 09, 2016.
588. S. Mulder and Z. Yaar. *The User Is Always Right: A Practical Guide to Creating and Using Personas for the Web*. New Riders, Berkeley, 2006.
589. P. Muller, P. Studer, and J. Bézivin. Platform independent web application modeling. In *UML 2003, Proceedings*, volume 2863 of *Lecture Notes in Computer Science*, pages 220–233. Springer, 2003.
590. R. Muller. *Database Design for Smarties: Using UML for Data Modeling*. Morgan Kaufman, San Francisco, 1999.
591. J. Müller-Brockmann. *Grid systems in graphic design*. Niggli, Sulgen/Zurich, 1996.
592. K. Mullet and D. Sano. *Designing visual interfaces*. Prentice-Hall, Englewood Cliffs, 1995.
593. G. L. Murphy. *The big book of concepts*. MIT Press, 2001.
594. D. Murray. Modelling for adaptivity. In *Proc. Mental Models and Human-Computer Interaction 2*, pages 81–95. North-Holland, 1991.
595. S. Murugesan. Web application development: Challenges and the role of web engineering. In *Web Engineering: Modelling and Implementing Web Applications*, pages 7–32. Springer, Berlin, 2008.
596. S. Murugesan and A. Ramanathan. *Web Personalisation - An Overview*, pages 65–76. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.
597. B. A. Myers. User interface software technology. *ACM Comput. Surv.*, 28(1):189–191, 1996.

598. B. A. Myers, S. E. Hudson, and R. F. Pausch. Past, present, and future of user interface software tools. *ACM Trans. Comput.-Hum. Interact.*, 7(1):3–28, 2000.
599. M. D. Myers and F. Tan. *Advanced topics in global information management*, chapter Beyond models of national culture in information systems research, pages 14–29. IGI Publishing, 2003.
600. J. Mylopoulos, A. Fuxman, and P. Giorgini. From entities and relationships to social actors and dependencies. In *Conceptual Modeling - ER 2000*, pages 27 – 36, Berlin, 2000. Springer.
601. J. Mylopoulos and R. Motschnig-Pitrik. Partitioning information bases with contexts. In *Proc. CoopIS '95*, pages 44–55. IEEE, 1995.
602. D. Navarre, P. Palanque, and M. Winckler. *Task Models and System Models as A Bridge Between Hci and Software Engineering*, pages 357–385. Springer London, London, 2009.
603. A. Navarro and T. Khan. *Effective Web Design*. Sybex, San Francisco, 1998.
604. M. Nebeling, S. Leone, and M. C. Norrie. Crowdsourced web engineering and design. In *ICWE*, volume 7387 of *Lecture Notes in Computer Science*, pages 31–45, 2012.
605. M. Nebeling and M. C. Norrie. Context-aware and adaptive web interfaces: A crowdsourcing approach. In *ICWE Workshops*, volume 7059 of *Lecture Notes in Computer Science*, pages 167–170, 2011.
606. U. Neisser. *Cognitive Psychology*. Prentice Hall, 1967.
607. B. Neumayr and M. S. und B. Thalheim. Modeling techniques for multi-level abstraction. In *The Evolution of Conceptual Modeling*, volume 6520 of *Lecture Notes in Computer Science*, pages 68–92, Berlin, 2011. Springer.
608. M. W. Newman and J. A. Landay. Sitemaps, storyboards, and specifications: A sketch of web site design practice. In *Proc. 3rd Conf. DIS*, pages 263–274. ACM, 2000.
609. P. Nicholson and B. Thalheim. Adaptation to learning styles. In *UNESCO-SAMEA conference*, 2004.
610. J. Nie and H. Hao. *The Information Architecture for Website Design*, pages 233–240. Springer US, Boston, MA, 2007.
611. J. Nielsen. *Designing Web Usability: The Practice of Simplicity*. New Riders Publishing, Indianapolis, 2000.
612. L. Nielsen. *Personas - User Focused Design*. Pearson Education, Springer, 2013.
613. T. Ninomiya and M. Mukaidono. *Fundamental Properties on Axioms of Kleene Algebra*, pages 311–320. Springer, Berlin, 2010.
614. S. Nishimura, A. Nevgi, and S. Tella. Communication style and cultural features in high/low context communication cultures: A case study of Finland, Japan and India. In *Proceedings of a Subject-Didactic Symposium*, Helsinki, Finland, 2008.
615. K. Nishino, N. Takata, Y. Iribe, S. Mizuno, K. Aoki, and Y. Fukumura. Developing a method of recommending e-learning courses based on students' learning preferences. In *KES (3)*, volume 6883 of *Lecture Notes in Computer Science*, pages 548–557, 2011.
616. R. Nixon. *Learning PHP, MySQL, JavaScript, and CSS - A Step-by-Step Guide to Creating Dynamic Websites*, 2nd Edition. O'Reilly, 2012.
617. K. Noack. Technologische und methodische Grundlagen von SCOPELAND. White paper, www.scopeland.de, 2009.

618. R. Noack. Generische Gestaltung von Web-Oberflächen. Master's thesis, Brandenburgische Technische Universität Cottbus, Institut für Informatik, 2005.
619. R. Noack. Rules and patterns for website orchestration. In *UNISCON*, pages 268–279, 2009.
620. R. Noack. *Content- und benutzungsgesteuertes generisches Layout: Screenography*. PhD thesis, Christian-Albrechts University of Kiel, Technical Faculty, Kiel, 2016.
621. R. Noack and B. Thalheim. Patterns for screenography. In *UNISCON*, pages 484–495, 2008.
622. U. Norbisrath, R. Jubeh, and A. Zündorf. *Story Driven Modeling*. CreateSpace Independent Publishing Platform, 2013.
623. D. Norman. *The design of everyday things*. Basic Books, New York, NY, 1988.
624. J. A. Nownes and J. A. Newmark. The information portfolios of interest groups: An exploratory analysis. *Interest Groups & Advocacy*, 5(1):57–81, 2016.
625. J. R. C. Nurse. *A business-oriented framework for enhancing web services security for e-business*. PhD thesis, University of Warwick, Coventry, UK, 2010.
626. T. A. O'Connell and E. D. Murphy. The usability engineering behind user-centered processes for web site development lifecycles. In P. Zaphiris, editor, *Human Computer Interaction Research in Web Design and Evaluation*, pages 1–21. Information Science Publishing, 2007.
627. A. Olivé. *Conceptual modeling of information systems*. Springer, Berlin, 2007.
628. T. W. Olle, J. Hagelstein, I. G. MacDonald, C. Rolland, and H. G. Sol. *Information systems methodologies - a framework for understanding (2. ed.)*. Addison-Wesley, 1991.
629. OMG. Metaobject facility (mof). <http://www.omg.org/mof/>, 2016. Accessed Dec. 01, 2016.
630. E. O'Neill and P. Johnson. Participatory task modelling: users and developers modelling users' tasks and domains. In *TAMODIA*, pages 67–74, 2004.
631. E. Ortner. *Methodenneutraler Fachentwurf*. B. G. Teubner Verlagsgesellschaft, Leipzig, 1997.
632. A. Ortony. *Metaphor and Thought*. Cambridge Press, Cambridge, 1979.
633. A. Osterwalder and Y. P. and L.C. Tucci. Clarifying business models: Origins, resent, and future. *Comm. AIS*, 16:1–25, 2004.
634. N. Ozkan, C. Paris, and . Balbo. *Understanding a Task Model: An Experiment*, pages 123–137. Springer London, London, 1998.
635. B. Paech. *Aufgabenorientierte Softwareentwicklung*. Springer, Berlin, 2000.
636. H. A. Parada Gélvez, A. Pardo, and C. D. Kloos. *Towards Combining Individual and Collaborative Work Spaces under a Unified E-Portfolio*, pages 488–501. Springer, Berlin, 2011.
637. J. Paredaens, P. De Bra, M. Gyssens, and D. Van Gucht. *The structure of the relational database model*. Springer, Berlin, 1989.
638. D. L. Parnas. Software Aging. In *ICSE: Proceedings of the 16th international conference on Software engineering*, pages 279–287, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.
639. S. Parumasur. The effect of organisational context on organisational development (OD) interventions. *SA Journal of Industrial Psychology*, 38(1), 2016. Retrieved February 1st, 2016.

640. O. Pastor and J. Molina. *Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling*. Springer, New York, New York, 2007.
641. F. Paterno. *Model-Based Design and Evaluation of Interactive Applications (Applied Computing)*. Springer, 1999.
642. M. Paulk, B. Curtis, M. Chrissis, and C. Weber. Capability maturity model for software, version 1.1. Technical Report CMU/SEI-93-TR-024, Software Engineering Institute, February 1993.
643. Z. Pawlak. Mathematical foundations of information retrieval. Technical Report CC PAS Reports 101, Warszawa, 1973.
644. T. Pencarelli and L. Gabbianelli. Intercultural aspect of customer management. In *17th Toulon-Verona International Conference - Excellence in Services*, pages 273–290, 2014.
645. S. Pepper and G. M. (eds.). Xml topic maps (xtm) 1.0. <http://www.topicmaps.org/xtm/1.0/>, 2001. TopicMaps.Org.
646. G. Persson. Meanings, models and metaphors; a study in lexical semantics in english. Umea Studies in the Humanities 92, Umea University, Imquist & Wiksell, 1990.
647. J. C. Peters. *Web information systems : a study of maintenance, change and flexibility*. PhD thesis, Brunel University London, UK, 2010.
648. C. J. Petrie. *Web Service Composition*. Springer, 2016.
649. A. Pickup. Next-generation web services: Enabling the customer-centric organisation. *Interactive Marketing*, 1(4):390–396, 2000.
650. K. Pipke. Verwendbarkeit von User-Interface-Metaphern in 3D-Welten unter spezieller Berücksichtigung von World-Wide-Web-Anwendungen. Master's thesis, FH Darmstadt, 1996.
651. T. Pittman and J. Peters. *The Art of Compiler Design: Theory and Practice*. Prentice Hall, Upper Saddle River, 1992.
652. T. Plumbaum, T. Stelter, and A. Korth. Semantic Web Usage Mining: Using Semantics to Understand User Intentions. In *UMAP*, volume 5535 of *Lecture Notes in Computer Science*, pages 391–396, 2009.
653. R. Poeschel and L. A. Kaluznin. *Funktionen- und Relationenalgebren: Ein Kapitel der diskreten Mathematik*. Birkhaeuser, Basel, 1979.
654. K. Pohl. *Requirements Engineering - Fundamentals, Principles, and Techniques*. Springer, 2010.
655. R. Polillo. Quality Models for Web [2.0] Sites: A Methodological Approach and a Proposal. In *ICWE Workshops*, volume 7059 of *Lecture Notes in Computer Science*, pages 251–265, 2011.
656. S. Polovina and W. Pearson. Communication + dynamic interface = better user experience. In C. Ghaoui, editor, *Encyclopedia of Human Computer Interaction*, pages 85–91. Idea Group Reference, 2006.
657. G. Polya. *How to solve it: A new aspect of mathematical method*. Princeton University Press, Princeton, 1945.
658. J. C. Preciado, M. L. Trigueros, R. Morales-Chaparro, F. Sánchez-Figueroa, G. Zhang, C. Kroiss, and N. Koch. Designing rich internet applications combining UWE and rux-method. In *Proceedings ICWE 2008*, pages 148–154. IEEE, 2008.
659. V. Propp. *Morphologie des Märchens*. Carl Hanser Verlag, München, 1972.
660. QVT. Query/view/transformation. <http://www.omg.org/spec/QVT/>, 2016. Accessed Dec. 01, 2016.

661. K. Radinsky, K. M. Svore, S. T. Dumais, J. Teevan, A. Bocharov, and E. Horvitz. Modeling and predicting behavioral dynamics on the web. In *WWW*, pages 599–608, 2012.
662. M. Rappa. Business models in the web. <http://digitalenterprise.org/models/models.pdf>, 2000.
663. A. Rasooli, P. Forbrig, and F. Taghiyareh. Collecting users profiles for web applications. In *HCSE*, volume 7623 of *Lecture Notes in Computer Science*, pages 275–282, 2012.
664. A. Rasooli, F. Taghiyareh, and P. Forbrig. Categorize web sites based on design issues. In *Proc. HCI 2011, Part IV*, volume 6764 of *Lecture Notes in Computer Science*, pages 510–519. Springer, 2011.
665. L. Razmerita, A. A. Angehrn, and A. Maedche. Ontology-based user modeling for knowledge management systems. In *User Modeling*, pages 213–217, 2003.
666. M. Reichert and B. Weber. *Enabling Flexibility in Process-Aware Information Systems. Challenges, Methods, Technologies*. Springer, Berlin, 2012.
667. K. Reinecke. Automatic adaptation of user interfaces to cultural preferences. *it - Information Technology*, 54(2):96–101, 2012.
668. K. Reinecke and A. Bernstein. Tell me where you've lived, and i'll tell you what you like: Adapting interfaces to cultural preferences. In *User Modeling, Adaptation, and Personalization (UMAP)*, volume 5535 of *Lecture Notes in Computer Science*, pages 185–196. Springer, 2009.
669. K. Reinecke and A. Bernstein. Improving performance, perceived usability, and aesthetics with culturally adaptive user interfaces. *ACM Transaction on Computer-Human Interaction*, 18(2):8, 2011.
670. K. Reinecke and A. Bernstein. Knowing what a user likes: A design science approach to interfaces that automatically adapt to the culture. *MIS Quarterly*, 37(2):427–453, 2013.
671. M. Reingruber and W. W. Gregory. *The data modeling handbook*. John Wiley & Sons, New York, 1994.
672. I. Reinhartz-Berger and D. Dori. OPM vs. UML - experimenting with comprehension and construction of web application models. *Empirical Software Engineering*, 10(1):57–80, 2005.
673. V. Repa. *Life Events: A Crucial Point of e-Government*, pages 33–47. Springer International Publ., Cham, 2015.
674. Rever. Dbmain resources including tutorials. <http://www.rever.eu/en/content/resources>, accessed Sept. 28, 2016, 2016.
675. F. Riaz-ud-Din, R. Noack, K.-D. Schewe, H. Ma, and B. Thalheim. Capturing forms in information systems design. In *4th International Conference on Innovations in Information Technology (Innovations'07)*, IIT, UAE, 2007. IEEE.
676. J. M. Rivero, G. Rossi, J. Grigera, E. R. Luna, and A. Navarro. From interface mockups to web application models. In *WISE*, pages 257–264, 2011.
677. J. N. Robbins. *Learning Web Design - A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics (4. ed.)*. O'Reilly, 2012.
678. S. Robertson and J. Robertson. *Requirements-led project management*. Pearson, Boston, 2005.
679. J. F. Roddick. Schema Evolution. In *Encyclopedia of Database Systems*, pages 2479–2481. Springer, 2009.
680. G. Roquier, R. Thavot, and M. Mattavelli. Methodology for the hardware/software co-design of dataflow programs. In *Proc. SiPS*, pages 174–179. IEEE, 2011.

681. L. Rosenfeld and P. Morville. *Information Architecture*. O'Reilly, Cambridge, 1998.
682. K. A. Ross. *View Adaptation*, pages 3324–3325. Springer US, Boston, MA, 2009.
683. G. Rossi, O. Pastor, D. Schwabe, and L. Olsina, editors. *Web Engineering: Modelling and Implementing Web Applications*. Springer, Berlin, 2008.
684. G. Rossi and D. Schwabe. Object-oriented design structures in web application models. *Ann. Software Eng.*, 13(1-4):97–110, 2002.
685. G. Rossi and D. Schwabe. Modeling and implementing web applications with OOHDM. In G. Rossi, O. Pastor, D. Schwabe, and L. Olsina, editors, *Web Engineering: Modelling and Implementing Web Applications*, pages 109–155. Springer, 2008.
686. G. Rossi, D. Schwabe, and F. Lyardet. Web application models are more than conceptual models. In P. C. et al., editor, *Advances in Conceptual Modeling*, volume 1727 of *LNCS*, pages 239–252. Springer-Verlag, Berlin, 1999.
687. O. Rostanin, B. Tschiedel, and B. Thalheim. Szenario-basiertes e-learning für adaptive Inhaltspräsentation. In K.-P. Jantke, W. Wittig, and J. Herrmann, editors, *Proc. LIT'2002*, pages 330–338. infix Publishers, 2002.
688. I. Rozanc and B. Slivnik. Producing the platform independent model of an existing web application. In *Proc. FedCSIS 2012*, pages 1341–1348, 2012.
689. D. Rupprecht, R. Blum, and K. Khakzar. Evaluation of web user interfaces for the online retail of apparel. In *HCI (9)*, volume 5618 of *Lecture Notes in Computer Science*, pages 74–83, 2009.
690. D. D. Ruscio, H. Muccini, and A. Pierantonio. A data-modelling approach to web application synthesis. *Int. J. Web Eng. Technol.*, 1(3):320–337, 2004.
691. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 2009.
692. I. Ruzsa. *Introduction to Metalogic*. Áron Publishers, Budapest, 1997.
693. J. Safra, I. Yeshua, and et. al. *Encyclopædia Britannica*. Merriam-Webster, 2003.
694. I. Saleh. *Formalizing Data-Centric Web Services*. Web-Scale Workflow and Analytics. Springer, 2015.
695. A. Samuel and J. Weir. *Introduction to Engineering: Modelling, Synthesis and Problem Solving Strategies*. Elsevier, Amsterdam, 2000.
696. B. Sargeant. Interactive storytelling: How picture book conventions inform multimedia book app narratives. *Austr. J. Intelligent Information Processing Systems*, 13(3), 2013.
697. S. Sasaki, P. Vojtás, K. Jannaschk, B. Thalheim, H. Jaakkola, and Y. Kiyoki. Multimedia information systems for social, cross-cultural and environmental computing. In P. Vojtás, Y. Kiyoki, H. Jaakkola, T. Tokuda, and N. Yoshida, editors, *Information Modelling and Knowledge Bases XXIV, 22nd European-Japanese Conference on Information Modelling and Knowledge Bases (EJC 2012), Prague, Czech Republic, June, 4-9, 2012*, volume 251 of *Frontiers in Artificial Intelligence and Applications*, pages 344–362. IOS Press, 2012.
698. K.-H. Saxer and P. A. Gloor. Navigation im hyperraum: Fisheye views in hypercard. In *Hypertext und Hypermedia: Von theoretischen Konzepten zur praktischen Anwendung*, volume 249 of *Informatik-Fachberichte*, pages 190–204. Springer, 1990.
699. J. M. Scandura and W. G. Roughead Jr. Conceptual organizers in short-term memory. *Journal of Verbal Learning and Verbal Behavior*, 6(4):679 – 682, 1967.

700. A.-W. Scheer. *Architektur integrierter Informationssysteme - Grundlagen der Unternehmensmodellierung*. Springer, Berlin, 1992.
701. G. Schellhorn. ASM refinement preserving invariants. *J. UCS*, 14(12):1929–1948, 2008.
702. B. Schewe. *Kooperative Softwareentwicklung - Ein objektorientierter Ansatz*. Deutscher Universitäts-Verlag, Wiesbaden, 1996.
703. B. Schewe and K.-D. Schewe. A user-centered method for the development of data-intensive dialogue systems: an object-oriented approach. In *Proc. IFIP Int. working conf. on information system concepts*, volume 26 of *IFIP Conference Proceedings*, pages 88–103. Chapman & Hall, 1995.
704. K.-D. Schewe. *The specification of data-intensive application systems*. Advanced PhD (Habilitation Thesis), Brandenburg University of Technology at Cottbus, Faculty of Mathematics, Natural Sciences and Computer Science, 1994.
705. K.-D. Schewe. Specification and development of correct relational database programs. Manuscript, TU Clausthal, 1996.
706. K.-D. Schewe. Design theory for advanced datamodels. In M. Orlowska and J. Roddick, editors, *Australasian Database Conference 2001*, <http://fims-www.massey.ac.nz/~kdschewe/publ.html>, 2001.
707. K.-D. Schewe. Querying web information systems. In H. S. Kunii, S. Jajodia, and A. Sølvberg, editors, *Conceptual Modeling - ER 2001, 20th International Conference on Conceptual Modeling*, volume 2224 of *Lecture Notes in Computer Science*, pages 571–584. Springer, 2001.
708. K.-D. Schewe. Support of integrated wireless web access through media types. In S. Spaccapietra, S. T. March, and Y. Kambayashi, editors, *Conceptual Modeling - (ER 2002 Workshops)*, volume 2503 of *Lecture Notes in Computer Science*, pages 173–181. Springer, 2002.
709. K.-D. Schewe. The power of media types. In X. Zhou, S. Y. W. Su, M. P. Papazoglou, M. E. Orlowska, and K. G. Jeffery, editors, *Web Information Systems (WISE 2004), 5th International Conference on Web Information Systems Engineering*, volume 3306 of *Lecture Notes in Computer Science*, pages 53–58. Springer, 2004.
710. K.-D. Schewe. Bargaining in e-business systems. In J. Akoka et al., editors, *Perspectives in Conceptual Modeling (ER 2005 Workshops)*, volume 3770 of *Lecture Notes in Computer Science*, pages 333–342. Springer, 2005.
711. K.-D. Schewe. The challenges in web information systems development in 15 pictures. In R. H. Kaschek, H. C. Mayr, and S. W. Liddle, editors, *Information Systems Technology and its Applications, 4th International Conference ISTA '2005*, volume 63 of *LNI*, pages 204–215. GI, 2005.
712. K.-D. Schewe, R. Kaschek, C. Wallace, and C. Matthews. Emphasizing the communication aspects for the successful development of electronic business systems. *Information Systems and E-Business Management*, 2005.
713. K.-D. Schewe, R. H. Kaschek, C. Matthews, and C. Wallace. Modelling web-based banking systems: Story boarding and user profiling. In S. Spaccapietra, S. T. March, and Y. Kambayashi, editors, *Conceptual Modeling - (ER 2002 Workshops)*, volume 2503 of *Lecture Notes in Computer Science*, pages 427–439. Springer, 2002.
714. K.-D. Schewe, Kinshuk, and T. Goh. Content adaptivity in wireless web access. *IJMC*, 2(3):260–270, 2004.

715. K.-D. Schewe and B. Schewe. View centered conceptual modelling – an object oriented approach. In B. Thalheim, editor, *Conceptual Modeling – ER '96*, LNCS 1157, pages 357–371. Springer, 1996.
716. K.-D. Schewe and B. Schewe. Integrating database and dialogue design. *Knowledge and Information Systems*, 2(1):1–32, 2000.
717. K.-D. Schewe and B. Thalheim. Fundamental concepts of object oriented databases. *Acta Cybernetica*, 11(4):49–81, 1993.
718. K.-D. Schewe and B. Thalheim. Readings in object-oriented databases. Reprint, BTU-Cottbus, accessible through <http://www.is.informatik.uni-kiel.de/~thalheim/OReport.pdf>, Collection of papers by C. Beeri, K.-D. Schewe, J.-W. Schmidt, D. Stemple, B. Thalheim, I. Wetzel, 1998.
719. K.-D. Schewe and B. Thalheim. Die Theorie der Medienobjekte für Internet-Dienste. 8. Leipziger Informatik-Tage, Leipzig (Germany), 2000. (in German).
720. K.-D. Schewe and B. Thalheim. Modeling interaction and media objects. In M. Bouzeghoub, Z. Kedad, and E. Métais, editors, *Natural Language Processing and Information Systems, 5th International Conference on Applications of Natural Language to Information Systems (NLDB 2000)*, volume 1959 of *Lecture Notes in Computer Science*, pages 313–324. Springer, 2001.
721. K.-D. Schewe and B. Thalheim. The co-design approach to WIS development in e-business and e-learning applications. In C. Bussler et al., editors, *Web Information Systems (WISE 2004 Workshops)*, volume 3307 of *Lecture Notes in Computer Science*, pages 181–189. Springer, 2004.
722. K.-D. Schewe and B. Thalheim. Reasoning about web information systems using story algebras. In G. Gottlob, A. A. Benczúr, and J. Demetrovics, editors, *Advances in Databases and Information Systems, 8th East European Conference (ADBIS 2004)*, volume 3255 of *Lecture Notes in Computer Science*, pages 54–66. Springer, 2004.
723. K.-D. Schewe and B. Thalheim. Structural media types in the development of data-intensive web information systems. In D. Taniar and W. Rahayu, editors, *Web Information Systems*, pages 34–70. IDEA Group, 2004.
724. K.-D. Schewe and B. Thalheim. Web information systems: Usage, content, and functionality modelling. Technical Report 2004-3, Christian Albrechts University Kiel, Institute of Computer Science and Applied Mathematics, Kiel, 2004.
725. K.-D. Schewe and B. Thalheim. An algorithmic approach to high-level personalisation of web information systems. In W. Fan, Z. Wu, and J. Yang, editors, *Advances in Web-Age Information Management, 6th International Conference (WAIM 2005)*, volume 3739 of *Lecture Notes in Computer Science*, pages 737–742. Springer, 2005.
726. K.-D. Schewe and B. Thalheim. The co-design approach to web information systems development. *IJWIS*, 1(1):5–14, 2005.
727. K.-D. Schewe and B. Thalheim. Conceptual modelling of web information systems. *Data Knowl. Eng.*, 54(2):147–188, 2005.
728. K.-D. Schewe and B. Thalheim. Tutorial 4: Reasoning about web information systems. In J. Akoka et al., editors, *Perspectives in Conceptual Modeling (ER 2005 Workshops)*, volume 3770 of *Lecture Notes in Computer Science*, pages 464–467. Springer, 2005.
729. K.-D. Schewe and B. Thalheim. Usage-based storyboard for web information systems. Technical Report 2006-13, Christian Albrechts University Kiel, Institute of Computer Science and Applied Mathematics, Kiel, 2006.

730. K.-D. Schewe and B. Thalheim. User models: A contribution to pragmatics of web information systems design. In K. Aberer, Z. Peng, E. A. Rundensteiner, Y. Zhang, and X. Li, editors, *Web Information Systems (WISE 2006)*, 7th International Conference on Web Information Systems Engineering, volume 4255 of *Lecture Notes in Computer Science*, pages 512–523. Springer, 2006.
731. K.-D. Schewe and B. Thalheim. Development of collaboration frameworks for web information systems. In *20th Int. Joint Conf. on Artificial Intelligence, Section EMC07 (Evolutionary models of collaboration)*, pages 27–32, Hyderabad, 2007.
732. K.-D. Schewe and B. Thalheim. Life cases: A kernel element for web information systems engineering. In J. Filipe and J. A. M. Cordeiro, editors, *Web Information Systems and Technologies, Third International Conference (WEBIST 2007)*, volume 8 of *Lecture Notes in Business Information Processing*, pages 139–156. Springer, 2007.
733. K.-D. Schewe and B. Thalheim. Life cases: An approach to address pragmatics in the design of web information systems. In J. Filipe, J. Cordeiro, B. Encarnacao, and V. Pedrosa, editors, *Proc. WebIST*, volume II (WIA), pages 5–12, 2007.
734. K.-D. Schewe and B. Thalheim. Personalisation of web information systems - A term rewriting approach. *Data Knowl. Eng.*, 62(1):101–117, 2007.
735. K.-D. Schewe and B. Thalheim. Pragmatics of storyboarding for web information systems: usage analysis. *IJWGS*, 3(2):128–169, 2007.
736. K.-D. Schewe and B. Thalheim. Storyboarding concepts for edutainment WIS. In H. Jaakkola, Y. Kiyoki, and T. Tokuda, editors, *Information Modelling and Knowledge Bases XIX, 17th European-Japanese Conference on Information Modelling and Knowledge Bases (EJC 2007)*, volume 166 of *Frontiers in Artificial Intelligence and Applications*, pages 59–78. IOS Press, 2007.
737. K.-D. Schewe and B. Thalheim. Term rewriting for web information systems - termination and church-rosser property. In B. Benatallah et al., editors, *Web Information Systems Engineering (WISE 2007)*, 8th International Conference on Web Information Systems Engineering, volume 4831 of *Lecture Notes in Computer Science*, pages 261–272. Springer, 2007.
738. K.-D. Schewe and B. Thalheim. Facets of media types. In R. Kaschek et al., editors, *Information Systems and e-Business Technologies, 2nd International United Information Systems Conference (UNISCON 2008)*, volume 5 of *Lecture Notes in Business Information Processing*, pages 296–305. Springer, 2008.
739. K.-D. Schewe and B. Thalheim. Web information systems co-design. In G. Kotisis et al., editors, *iiWAS'2008 - The Tenth International Conference on Information Integration and Web-based Applications Services*, page 3. ACM, 2008.
740. K.-D. Schewe and B. Thalheim. Web information systems portfolios: A contribution to pragmatics. In J. Filipe and J. Cordeiro, editors, *Web Information Systems and Technologies – 6th International Conference (WEBIST 2010)*, volume 75 of *Lecture Notes in Business Information Processing*, pages 147–161. Springer, 2010.
741. K.-D. Schewe and B. Thalheim. *Correct Software in Web Applications and Web Services*, chapter Co-Design of Web Information Systems, pages 293–332. Texts & Monographs in Symbolic Computation. Springer, Wien, 2015.
742. K.-D. Schewe, B. Thalheim, A. Binemann-Zdanowicz, R. H. Kaschek, T. Kuss, and B. Tschiedel. A conceptual view of web-based e-learning systems. *Education and Information Technologies*, 10(1-2):83–110, 2005.

743. K.-D. Schewe, B. Thalheim, and R. H. Kaschek. A deontic logic for group-oriented web information systems. In F. Feltz, A. Oberweis, and B. Otjacques, editors, *EMISA 2004, Informationssysteme im E-Business und E-Government*, volume P-56 of *Lecture Notes in Informatics*, pages 107–116. GI, 2004.
744. K.-D. Schewe, B. Thalheim, and A. Tretiakov. Personalised web-based learning systems. In *Proceedings of the 5th IEEE International Conference on Advanced Learning Technologies, ICALT 2005, Kaohsiung, Taiwan, July 5-8, 2005*, pages 655–658. IEEE Computer Society, 2005.
745. K.-D. Schewe, B. Thalheim, and Q. Wang. Customising web information systems according to user preferences. *World Wide Web*, 12(1):27–50, 2009.
746. K.-D. Schewe, B. Thalheim, and J. Zhao. Quality assurance in web information systems development. In *QSIC 2007*, pages 219–224. IEEE, 2007.
747. K.-D. Schewe, B. Thalheim, and S. Zlatkin. Modelling actors and stories in web information systems. In A. E. Doroshenko et al., editors, *Third International Conference on Information Systems Technology and its Applications (ISTA 2004)*, volume P-48 of *Lecture Notes in Informatics*, pages 13–23, Salt Lake City (Utah), 2004. GI.
748. K.-D. Schewe and Q. Wang. A formal model for service mediators. In J. Trujillo, G. Dobbie, H. Kangassalo, S. Hartmann, M. Kirchberg, M. Rossi, I. Reinhartz-Berger, E. Zimányi, and F. Frasincar, editors, *Advances in Conceptual Modeling - Applications and Challenges, (ER 2010 Workshops)*, volume 6413 of *Lecture Notes in Computer Science*, pages 76–85. Springer, 2010.
749. K.-D. Schewe and Q. Wang. Preferential refinements of abstract state machines for service mediators. In A. Tang and H. Muccini, editors, *12th International Conference on Quality Software (QSIC 2012)*, pages 158–166. IEEE, 2012.
750. K.-D. Schewe and Q. Wang. What constitutes a service on the web? In *Correct Software in Web Applications and Web Services*, pages 257–292. Springer, 2015.
751. K.-D. Schewe, J. Zhao, and B. Thalheim. Quality assurance in web information systems development. In *Seventh International Conference on Quality Software (QSIC 2007)*, pages 219–224. IEEE Computer Society, 2007.
752. G. Schmalz. High context low context. Available in <http://www.giselaschmalz.com/cultures-and-context-edward-t-hall/>, 2016. Retrieved October 19th, 2016.
753. J. Schmidt and M. Brodie, editors. *Relational Database Systems: Analysis and Comparison*. Springer, Heidelberg, 1983.
754. J. Schmidt and H.-W. Sehring. Conceptual content modeling and management - the rationale of an asset language. In *Proc. PSI'03, LNCS , Springer, 2003*, 2003. Perspectives of System Informatics.
755. J. W. Schmidt and H.-W. Sehring. Dockets: a model for adding vaulue to content. In *Proc. ER'99*, volume 1728 of *LNCS*, pages 248–262, 1999.
756. P. Schmidt and B. Thalheim. Management of UML clusters. In J.-R. Abrial and U. Glässer, editors, *Rigourous Methods for Software Construction and Analysis*, LNCS 5115, pages 110–128, Berlin, 2009. Springer.
757. T. Schmidt. Requirements, concepts, and solutions for the development of a basic technology of information services - the client. Master's thesis, Brandenburg University of Technology at Cottbus, Institute of Computer Science, 1998. In German.
758. M. Schrefl and M. Stumptner. Behavior consistent refinement of object life cycles. In *ER*, volume 1331 of *Lecture Notes in Computer Science*, pages 155–168. Springer, 1997.

759. H. Schreiber, K.-E. Sommerfeld, and G. Starke. *Deutsche Wortfelder für den Sprachunterricht: Verbgruppen*. VEB Verlag Enzyklopädie, Leipzig, 1990.
760. F. Schumann. Beiträge zur Analyse der Gesichtswahrnehmungen. Erste Abhandlung: Einige Beobachtungen über die Zusammenfassung von Gesichtseindrücken zu Einheiten. *Zeitschrift für Psychologie und Physiologie der Sinnesorgane*, 23:1–23, 1900.
761. D. Schwabe and G. Rossi. The object-oriented hypermedia design model. *Communications of the ACM*, 38(8):45–46, 1995.
762. D. Schwabe and G. Rossi. An object oriented approach to web-based application design. *TAPOS*, 4(4):207–225, 1998.
763. D. Schwabe, G. Rossi, and S. Barbosa. Systematic hypermedia design with OOHDM. In *Proc. Hypertext '96*, pages 116–128. ACM Press, 1996.
764. R. L. Schwartz and T. Phoenix. *Learning Perl - making easy and hard things possible* (3. ed.). O'Reilly, 2001.
765. R. Schwietzke. Requirements, concepts, and solutions for the development of a basic technology of information services - the server. Master's thesis, Brandenburg University of Technology at Cottbus, Institute of Computer Science, 1998. In German.
766. W. Schwinger, W. Retschitzegger, A. Schauerhuber, G. Kappel, M. Wimmer, B. Pröll, C. Cachero, S. Casteleyn, O. De Troyer, P. Fraternali, I. Garrigós, F. Garzotto, A. Ginige, G.-J. Houben, N. Koch, N. Moreno, O. Pastor, P. Paolini, V. Pelechano, G. Rossi, D. Schwabe, M. Tisi, A. Vallecillo, K. van der Sluijs, and G. Zhang. A survey on web modeling approaches for ubiquitous web applications. *IJWIS*, 4(3):234–305, 2008.
767. J. Searle and D. Vanderveken. *Foundations of illocutionary logic*. Cambridge University Press, Cambridge, 1985.
768. J. R. Searle. Meaning and speech acts. *Philosophical Review*, 71:423–432, 1962.
769. J. R. Searle. *Expression and Meaning: Studies in the Theory of Speech Acts*. Cambridge University Press, 1979.
770. R. W. Sebesta. *Programming the World Wide Web*. Addison-Wesley, Boston, MA, 2006.
771. N. M. Seel, editor. *Didactics*, pages 985–985. Springer US, Boston, MA, 2012.
772. L. Seger. *Making a good script*. Samuel French, Hollywood, 1994.
773. L. Seger. *Vom Buch zum Drehbuch*. Zweitausendundeins, Frankfurt a.M., 2001.
774. Classic software architecture definitions. Software Engineering Institute at Carnegie Mellon University, see <http://www.sei.cmu.edu/architecture/start/classicdefs.cfm>, 2016, accessed Sept. 29, 2016.
775. G. Semper. *Die vier Elemente der Baukunst*. Braunschweig, 1851.
776. D. Shafer and K. Yank. *Cascading stylesheets - anspruchsvolle Websites mit CSS gestalten: Grundlagen, Designtechniken und Referenz*. dpunkt.verlag, 2004.
777. H. Sharp, Y. Rogers, and J. Preece. *Interaction Design: Beyond Human-Computer Interaction*. Wiley, Chichester, 2007.
778. D. E. Shasha. *Database tuning-a principle approach*. Prentice-Hall, Englewood Cliffs, 1992.
779. M. Shaw, editor. *E-Business Management - Integration of Web Technologies with Business Models*. Springer, 2002.
780. M. Shaw and D. Garlan. *Software architectures: Perspectives of an emerging discipline*. Prentice-Hall, Upper Saddle River, 1996.

781. B. Shneiderman. *Designing the user interface*. Addison-Wesley, NY, 1997.
782. B. Shneiderman and C. Plaisant. *Designing the User Interface: Strategies for effective human-computer interaction*. Addison-Wesley, 2004.
783. J. Siedersleben. *Moderne Softwarearchitektur*. dpunkt-Verlag, Heidelberg, 2004.
784. D. Siegel. *Web site design: Killer web sites*. Markt und Technik, München, 1997.
785. D. Siegel. *The secrets of successful web sites*. Markt und Technik, München, 1998.
786. L. Silverston. *The data model resource book. A library of universal data models by industry types*, volume 2. Wiley, 2001.
787. L. Silverston. *The data model resource book. Revised edition*, volume 1. Wiley, 2001.
788. L. Silverston and P. Agnew. *The data model resource book. Universla pattern of data modeling*, volume 2. Wiley, 2009.
789. G. Simsion. *Data modeling - Theory and practice*. Technics Publications, LLC, New Jersey, 2007.
790. G. Simsion and G. Witt, editors. *Data modelling essentials*. Morgan Kaufmann, San Francisco, 2005.
791. R. Singh and A. Solar-Lezama. SPT: storyboard programming tool. In *Proc. CAV*, volume 7358 of *Lecture Notes in Computer Science*, pages 738–743. Springer, 2012.
792. D. Skopec. *Digital Layout for the Internet and other Media*. Ava Publishing SA, 2003.
793. M. Skusa. *Semantische Kohärenz in der Softwareentwicklung*. PhD thesis, CAU Kiel, 2011.
794. M. Skusa and B. Thalheim. *Wissenschaft und Kunst der Modellierung: Modelle, Modellieren, Modellierung*, chapter Kohärente Multi-Modell-Entwicklung, pages 431–454. De Gryuter, Boston, 2015.
795. R. Slavin. *Cooperative Learning: Theory, research, and practice*. Allyn and Bacon, Boston, 1995.
796. J. Sonnberger and A. Binemann-Zdanowicz. KOPRA - ein adaptives Lehr-Lernsystem für kooperatives Lernen. In *GMW'2004, Graz, Austria, Sept. 2004*, pages 274–285, 2004.
797. K. Sousa, A. Schilling, and E. Furtado. Integrating usability, semiotic, and software engineering. In A. Dasso and A. Funes, editors, *Verification, Validation and Testing in Software Engineering*, pages 47–70. Idea Group Publishing, 2007.
798. J. F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.
799. J. F. Sowa. *Knowledge Representation*. Brooks/Cole, Pacific Grove (California), USA, 2000.
800. E. Söztutar, I. H. Toroslu, and M. A. Bayir. Semantically enriched event based model for web usage mining. In *WISE*, 2010.
801. L. Spillmann. Gehirn und Gestalt. *Kognitionswissenschaft*, 9(3):122–143, 2001.
802. S. Srinivasa. *A Calculus of Fixed-Points for Characterising Interactive Behaviour of Information Systems*. PhD thesis, BTU Cottbus, Fachbereich Informatik, Cottbus, 2001.
803. T. Stahl and M. Völter. *Model-driven software architectures*. dPunkt, Heidelberg, 2005. (in German).

804. T. Stapelkamp. *Screen- und Interfacedesign*. Springer, 2007.
805. R. Statkaityte. Culture sensitivity in software processes. Master's thesis, Information Technology (Pori). Tampere, Tampere University of Technology, <http://dspace.cc.tut.fi/dpub/handle/>, 2011.
806. R. Statkaityte and H. Jaakkola. Modeling the multicultural issues in software engineering processes. In *Information Modelling and Knowledge Bases*, volume XXV of *Frontiers in Artificial Intelligence and Applications*, 260, pages 190–199. IOS Press, 2014.
807. M. Stefik. *Internet Dreams; Archetypes, Myths, and Metaphors*. The MIT Press, 1997.
808. Z. Stojanovic, A. N. W. Dahanayake, and H. G. Sol. A service-oriented component modeling approach. In *Information Modeling Methods and Methodologies*, pages 300–322. Idea Group, 2005.
809. J. Strutz and G. Degel. Offene übungsaufgaben und Praktika im e-Learning. Einbindung, Auswertung und Bewertung im Tutorsystem DaMiT. In *LIT'02*, pages 410 – 420. infix, 2002.
810. D. Suciu and L. Wong. On two forms of structural recursion. In *Proc. ICDT'95*, volume 893 of *Lecture Notes in Computer Science*, pages 111–124. Springer, 1995.
811. T. Suhardijanto, Y. Kiyoki, and A. R. Barakbah. A culture-dependent metadata creation method for color-based impression extraction with cultural color spaces. In *Information Modelling and Knowledge Bases XXII, 20th European-Japanese Conference on Information Modelling and Knowledge Bases (EJC 2010), Jyväskylä, Finland, 31 May - 4 June 2010*, volume 225 of *Frontiers in Artificial Intelligence and Applications*, pages 333–343. IOS Press, 2010.
812. K. Sumi and M. Nagata. Persuasive narrative via digital storytelling. In *Proc. HCI, Part I*, volume 8016 of *Lecture Notes in Computer Science*, pages 276–283. Springer, 2013.
813. M. Sutherland and N. A. M. Maiden. Storyboarding requirements. *IEEE Software*, 27(6):9–11, 2010.
814. C. Tacitus. A dialogue concerning oratory, or the causes of corrupt eloquence. The Gutenberg Project. The works of Cornelius Tacitus. , Vol. VIII. <http://www.gutenberg.org/files/15017/15017-h/15017-h.htm>, 2016. Accessed, March 10, 2016.
815. Y. Tanaka. *Meme media and meme market architectures: Knowledge media for editing, distributing, and managing intellectual resources*. J. Wiley, Hoboken, 2003.
816. G. Teege. Object-oriented activity support: A model for integrated cscw systems. *Computer Supported Cooperative Work*, 5(1):93–124, 1996.
817. Q. Teng. *Structural biology*. Springer, 2005.
818. M. Teodorakis, A. Analyti, P. Constantopoulos, and N. Spyros. Context in information bases. In *Proceedings CoopIS '98*, pages 260–270, 1998.
819. M. Teodorakis, A. Analyti, P. Constantopoulos, and N. Spyros. Querying contextualized information bases. In *Proceedings ICT & P '99*, 1999.
820. T. J. Teorey, S. Lightstone, and T. Nadeau. *Database modeling and design*. Morgan Kaufmann, San Francisco, 2006.
821. B. Thalheim. Design tools for large relational database systems. In *MFDBS*, volume 305 of *Lecture Notes in Computer Science*, pages 210–224. Springer, 1987.

822. B. Thalheim. *Dependencies in relational databases*. Teubner, Leipzig, 1991.
823. B. Thalheim. Fundations of entity-relationship modeling. *Annals of Mathematics and Artificial Intelligence*, 7(1-4):197–256, 1993.
824. B. Thalheim. Codesign of database systems and interaction - thin and consistent uml. In *OTS'2000*, pages 1–17, Maribor, 2000.
825. B. Thalheim. *Entity-Relationship Modeling: Foundations of Database Technology*. Springer-Verlag, 2000.
826. B. Thalheim. The person, organization, product, production, ordering, delivery, invoice, accounting, budgeting and human resources pattern in database design. Technical Report Preprint I-07-2000, Brandenburg University of Technology at Cottbus, Institute of Computer Science, 2000.
827. B. Thalheim. Readings in fundamentals of interaction in information systems. Reprint, BTU-Cottbus, accessible through <http://www.is.informatik.uni-kiel.de/~thalheim/interaction.pdf>, Collection of papers by C. Binder, W. Clauß, A. Düsterhöft, T. Feyer, T. Gutacker, B. Heinze, J. Lewerenz, M. Roll, B. Schewe, K.-D. Schewe, K. Seelig, S. Srinivasa, B. Thalheim, 2000.
828. B. Thalheim. Component construction of database schemes. In *Proc. ER'02, LNCS 2503*, pages 20–34. Springer, 2002.
829. B. Thalheim. Database component ware. In *ADC'2003*, pages 13–26. Australian Computer Science Communications, Vol 25, Number 2, 2003.
830. B. Thalheim. Informationssystem-Entwicklung. In *BTU Cottbus, Computer Science Institute, Technical Report I-15-2003*, Cottbus, 2003.
831. B. Thalheim. Application development based on database components. In Y. K. H. Jaakkola, editor, *EJC'2004*, Information Modeling and Knowledge Bases XVI, pages 28–45. IOS Press, 2004.
832. B. Thalheim. The co-design framework to content specification. In W. Abramowicz, editor, *BIS'2004*, pages 326–351. IEEE Press, 2004.
833. B. Thalheim. Co-design of structuring, functionality, distribution, and interactivity for information systems. In S. Hartmann and J. F. Roddick, editors, *Conceptual Modelling 2004, First Asia-Pacific Conference on Conceptual Modelling (APCCM2004)*, volume 31 of *CRPIT*, pages 3–12. Australian Computer Society, 2004.
834. B. Thalheim. Component development and construction for database design. *Data and Knowledge Engineering*, 54:77–95, 2005.
835. B. Thalheim. The conceptual framework to user-oriented content management. In *EJC'06*, Trojanovice, May 2006.
836. B. Thalheim. Engineering database component ware. In D. Draheim and G. Weber, editors, *Trends in Enterprise Application Architecture, 2nd International Conference, TEAA 2006, Berlin, Germany, November 29 - December 1, 2006, Revised Selected Papers*, volume 4473 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2006.
837. B. Thalheim. The conceptual framework to user-oriented content management. In *Information Modelling and Knowledge Bases*, volume XVIII of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2007.
838. B. Thalheim. Extended entity-relationship model. In *Encyclopedia of Database Systems*, pages 1083–1091. Springer US, 2009.
839. B. Thalheim. Model suites for multi-layered database modelling. In *Information Modelling and Knowledge Bases XXI*, volume 206 of *Frontiers in Artificial Intelligence and Applications*, pages 116–134. IOS Press, 2010.

840. B. Thalheim. The enhanced entity-relationship model. In *The Handbook of Conceptual Modeling: Its Usage and Its Challenges*, chapter 12, pages 165–208. Springer, Berlin, 2011.
841. B. Thalheim. The science and art of conceptual modelling. *Trans. Large-Scale Data- and Knowledge-Centered Systems VI*, 7600:76–105, 2012.
842. B. Thalheim. Syntax, semantics and pragmatics of conceptual modelling. In *NLDB*, volume 7337 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2012.
843. B. Thalheim. Open problems of information systems research and technology. In *Invited Keynote, BIR'2013. LNBIB 158*, pages 10–18. Springer, 2013.
844. B. Thalheim. Web information systems, their development, specification, technology, Web 1.0, Web 2.0, Web 3.0. Reprint, CAU Kiel, accessible through <http://www.is.informatik.uni-kiel.de/~thalheim/WIS.pdf>. Collection of papers by S. Al-Fedaghi, A. Bienemann, A. Berztiss, E. Börger, J. Demetrovics, A. Düsterhöft, T. Feyer, G. Grieser, P. Grigoriev, H. Jaakkola, K.-P Jancke, R. Kaschek, M. Kirchberg, U. Krautz, T. Kuss, L. Landwehr, H. Ma, T. Mäkinen, M. Memmel, A. Molnar, T. Moritz, P. Nicholson, R. Noack, K. Odey, T. Raak, I. Romalis, O. Rostanin, K.-D. Schewe, T. Schwanzara-Benoit, P. Schmidt, J. Sonnberger, S. Srinivasa, B. Thalheim, B. Tschiedel, T. Varkoi, and L. Zhang, 2013.
845. B. Thalheim. The pragmatic notion of information. *Annales Univ. Sci. Budapest., Sect. Comp.*, 43:69–87, 2014.
846. B. Thalheim. Model capsules for research and engineering networks. In *New Trends in Databases and Information Systems*, volume 637 of *CCIS*, pages 189–200. Springer, 2016.
847. B. Thalheim, S. S. Al-Fedaghi, and K. Al-Saqabi. Information stream based model for organizing security. In *ARES*, pages 1405–1412. IEEE Computer Society, 2008.
848. B. Thalheim and A. Dahanayake. A conceptual model for services. In *Invited Keynote, CMS 2015, ER 2015 workshop*, LNCS 9382, pages 51–61, Berlin, 2015. Springer.
849. B. Thalheim and A. Düsterhöft. The use of metaphorical structures for internet sites. *Data & Knowledge Engineering*, 35:161–180, 2000.
850. B. Thalheim and A. Düsterhöft. SiteLang: Conceptual modeling of internet sites. In H. S. Kunii, S. Jajodia, and A. Sølvberg, editors, *Conceptual Modeling – ER 2001*, volume 2224 of *LNCS*, pages 179–192. Springer-Verlag, Berlin, 2001.
851. B. Thalheim and H. Jaakkola. The cultural background and support for smart web information systems. In *AHA'2016*, LNCS 9975, pages 164–183, Cham, 2016. Springer International Publ.
852. B. Thalheim, H. Jaakkola, T. Nakanishi, S. Sasaki, and K.-D. Schewe. Conceptual modelling of collaboration for information systems. In T. Tokuda, Y. Kiyoki, H. Jaakkola, and N. Yoshida, editors, *Information Modelling and Knowledge Bases XXV, 23rd European-Japanese Conference on Information Modelling and Knowledge Bases (EJC 2013)*, volume 260 of *Frontiers in Artificial Intelligence and Applications*, pages 272–305. IOS Press, 2013.
853. B. Thalheim and T. Kobienia. Generating db queries for web nl requests using schema information and db content. In *NLDB2001*, pages 205–209, 2001.
854. B. Thalheim and I. Nissen, editors. *Wissenschaft und Kunst der Modellierung*. De Gruyter, Ontos Verlag, Berlin, 2015.

855. B. Thalheim and I. Nissen. *Wissenschaft und Kunst der Modellierung: Modelle, Modellieren, Modellierung*, chapter Fallstudien zum Modellbegriff, pages 549–602. De Gruyter, Boston, 2015.
856. B. Thalheim, K.-D. Schewe, and H. Ma. Conceptual application domain modelling. In *APCCM*, volume 96 of *CRPIT*, pages 49–57. Australian Computer Society, 2009.
857. B. Thalheim, K.-D. Schewe, A. Prinz, and B. Buchberger, editors. *Correct Software in Web Applications and Web Services*. Springer, 2015.
858. B. Thalheim, K.-D. Schewe, I. Romalis, T. Raak, and G. Fiedler. Website modeling and website generation. In Springer, editor, *ICWE'2004*, LNCS 3140, pages 577–578, 2004.
859. B. Thalheim, M. Tropmann-Frick, and T. Ziebermayr. Application of generic workflows for disaster management. In *Information Modelling and Knowledge Bases*, volume XXV of *Frontiers in Artificial Intelligence and Applications*, 260, pages 64–81. IOS Press, 2014.
860. M. Theodorakis, A. Analyti, P. Constantopoulos, and N. Spyros. Contextualization as an abstraction mechanism for conceptual modelling. In *Conceptual Modeling – Proc. ER'99*, volume 1728 of *LNCS*, pages 475–489. Springer-Verlag, 1999.
861. D. Theodoratos and T. Sellis. Data warehouse schema and instance design. In *Conceptual Modeling – ER'98*, volume 1507 of *LNCS*, pages 363–376. Springer-Verlag, 1998.
862. N. Thompson. *An Introduction to Portfolio Analysis*, pages 1–3. Palgrave Macmillan UK, London, 1993.
863. K. Thorn. The art of storyboarding. *eLearn Magazine*, 2011(8):7, 2011.
864. P. Timmers. Business models for the electronic market. *Electronic Markets*, 8(2):3–8, 1998.
865. S. Torge, W. Esswein, S. Lehrmann, and B. Thalheim. Categories for description of reference models. In *Information Modelling and Knowledge Bases*, volume XXV of *Frontiers in Artificial Intelligence and Applications*, 260, pages 229–240. IOS Press, 2014.
866. I. Torre. Goals, tasks and application domains as the guidelines for defining a framework for user modelling. In *User Modeling*, volume 2109 of *Lecture Notes in Computer Science*, pages 260–264, 2001.
867. H. Trætteberg. *UI Design without a Task Modeling Language – Using BPMN and Diamodl for Task Modeling and Dialog Design*, pages 110–117. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
868. F. Trompenaars and C. Hampden-Turner. *Riding the waves of culture*. McGraw-Hill, New York, 1998.
869. M. Tropmann. Funktionale Integration heterogener Datenbanken. Master's thesis, CAU Kiel, Institut für Informatik, Kiel, Januar 2008.
870. M. Tropmann and B. Thalheim. Performance forecasting for performance critical huge databases. In *Information Modelling and Knowledge Bases*, volume XXII, pages 206–225. IOS Press, 2011.
871. M. Tropmann and B. Thalheim. Mini story composition for generic workflows in support of disaster management. In *DEXA 2013*, pages 36–40. IEEE Computer Society, 2013.
872. M. Tropmann-Frick. *Genericity in Process-Aware Information Systems*. PhD thesis, Christian-Albrechts University of Kiel, Technical Faculty, Kiel, 2016.

873. M. Tropmann-Frick, B. Thalheim, D. Leber, G. Czech, and C. Liehr. Generic workflows - a utility to govern disastrous situations. In *Information Modelling and Knowledge Bases*, volume XXVI of *Frontiers in Artificial Intelligence and Applications*, 272, pages 417–428. IOS Press, 2014.
874. H. L. Truong, M. Comerio, A. Maurino, S. Dustdar, F. D. Paoli, and L. Panziera. On identifying and reducing irrelevant information in service composition and execution. In *WISE*, pages 52–66, 2010.
875. K. N. Truong, G. R. Hayes, and G. D. Abowd. Storyboarding: an empirical determination of best practices and effective guidelines. In *Proc. Conf. on Designing Interactive Systems*, pages 12–21. ACM, 2006.
876. C. H. Tsang, C. S. Lau, and Y. K. Leung. *Object-Oriented Technology: From Diagram to Code with Visual Paradigm for UML*. McGraw-Hill, 2010.
877. J. Tschichold. *Die neue Typografie*. Verlag des Bildungsverbandes der Deutschen Buchdrucker, 1928.
878. B. Tschiedel, A. Binemann-Zdanowicz, and B. Thalheim. Logistics for learning objects. In *Proc. IFIP TC3 eTRAIN'2003*, volume 167 of *IFIP International Federation for Information Processing*, pages 315–323. Springer, 2005.
879. M. M. Tuffield. *Telling your story : autobiographical metadata and the semantic web*. PhD thesis, University of Southampton, UK, 2010.
880. E. Tufte. *Visual explanations*. Graphics Press, 1997.
881. T. Uramoto. Canonical finite models of kleene algebra with tests. *J. Log. Algebr. Meth. Program.*, 85(4):595–616, 2016.
882. D. Ursino. *Extraction and exploitation of intentional knowledge from heterogeneous resources*. Number 558 in LNCS. Springer, 2002.
883. UWE. Magicuwe. http://uwe.pst_ifi.lmu.de/toolMagicUWE.html, 2016. Accessed Dec. 05, 2016.
884. UWE. Metamodel and profile. http://uwe.pst_ifi.lmu.de/publicationsMetamodelAndProfile.html, 2016. Accessed Dec. 09, 2016.
885. K. Väänänen. *Metaphor-Based User Interfaces for Information Authoring, Visualization and Navigation in Multimedia Environments*. PhD thesis, TH Darmstadt, 1995.
886. P. Valderas and V. Pelechano. Introducing requirements traceability support in model-driven development of web applications. *Information & Software Technology*, 51(4):749–768, 2009.
887. E. Vale. *The technique of screen and television writing*. Simon and Schuster, New York, 1982.
888. A. Vallecillo, N. Koch, C. Cachero, S. Comai, P. Fraternali, I. Garrigós, J. Gómez, G. Kappel, A. Knapp, M. Matera, S. Meliá, N. Moreno, B. Pröll, T. Reiter, W. Retschitzegger, J. E. Rivera, A. Schauerhuber, W. Schwinger, M. Wimmer, and G. Zhang. MDWENet: A practical approach to achieving interoperability of model-driven web engineering methods. In *MDWE*, 2007.
889. F. Valverde, P. Valderas, and J. Fons. OOWS suite: Un entorno de desarrollo para aplicaciones web basado en MDA. In *Proc. (CIBSE 2007)*, pages 253–266, 2007.
890. J. Van den Bussche. Complex object multi-level fixpoint queries. In *MFDDBS 91,,* volume 495 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 1991.
891. J. Van den Bussche and D. Van Gucht. Non-deterministic aspects of object-creating database transformations. In *Proc. Foundations of Models and Lan-*

- guages for Data and Objects*, Workshops in Computing, pages 3–16. Springer, 1992.
892. C. van der Lelie. The value of storyboards in the product design process. *Personal and Ubiquitous Computing*, 10(2-3):159–162, 2006.
 893. D. K. Van Duyne, J. A. Landay, and J. I. Hong. *The Design of Sites*. Addison-Wesley, Boston, 2002.
 894. R. Vdovjak, F. Frasincar, G.-J. Houben, and P. Barna. Engineering semantic web information systems in Hera. *Journal of Web Engineering*, 2(1-2):3–26, 2003.
 895. J. Venable. The role of theory and theorising in design science research. In *DESIST*, 2006.
 896. J. R. Venable. *Five and Ten Years on: Have DSR Standards Changed?*, pages 264–279. Lecture Notes in Computer Science 9073. Springer International Publishing, Cham, 2015.
 897. V. Vestenicky. *Schema integration as view cooperation*. PhD thesis, Charles University Prague, Computer Science, 2005.
 898. R. D. Virgilio, R. Torlone, and G. Houben. Rule-based adaptation of web information systems. *World Wide Web*, 10(4):443–470, 2007.
 899. M. von Rosing, A. Scheer, J. A. Zachman, D. T. Jones, J. P. Womack, and H. von Scheel. Phase 3: Process concept evolution. In *The Complete Business Process Handbook: Body of Knowledge from Process Modeling to BPM, Volume I*, pages 37–77. Morgan Kaufmann/Elsevier, 2015.
 900. M. von Rosing, A.-W. Scheer, H. von Scheel, A. D. M. Svendsen, A. Kokkonen, A. M. Ross, A. F. Bøgebjerg, A. Olsen, A. Dicks, A. Q. Gill, B. Bach, B. J. Storms, C. Smit, C. Clemmensen, C. K. Swierczynski, C. Utschig-Utschig, D. Moorcroft, D. T. Jones, D. Coloma, D. Boykin, D. H. Muhita, D. Gonçalves, F. M. Maggi, F. Zhao, F. Senghore, F. Dandashi, F. Cummins, F. Stoffel, G. von Scheel, G. von Rosing, G. Doucet, G. Meiling, G. O. Jansson, H. Scheruhn, H. Bohn, H. de Man, H. Kuil, H. N. Vester, J. Gammelgaard, J. P. Womack, J. W. Ross, J. Greer, J. T. Nielsen, J. A. Zachman, J. Bertram, J. Golden, J. M. Rogers, J. Erasmus, J. von Scheel, and J. Waters. Business process trends. In *The Complete Business Process Handbook: Body of Knowledge from Process Modeling to BPM, Volume I*, pages 187–216. Morgan Kaufmann/Elsevier, 2015.
 901. R. Walter. *Essentials of Screenwriting: The Art, Craft, and Business of Film and Television Writing*. Plume, 2010.
 902. T. Walter. *Kompendium der Web-Programmierung - Dynamische Web-Site*. X.media.press, Berlin, 2008.
 903. H. Wang, J. Bian, Q. Wu, and Y. Wang. itucome: Hdcfg-based incremental tuning HW/SW co-design methodology for multi-level exploration. In *Proc. CSCWD*, pages 978–983. IEEE Computer Society, 2005.
 904. Q. Wang and B. Thalheim. Data migration: A theoretical perspective. *DKE*, 87:260–278, 2013.
 905. H. Weber. *Continuous Engineering of Information and Communication Infrastructures*, pages 22–29. Springer, Berlin, 1999.
 906. WebML: The web modelling languages. www., Accessed Sept. 17, 2016 2016.
 907. WebRatio. Webratio platform. <http://www.webratio.com/site/content/en/web-application-development>, 2016. Accessed Dec. 09, 2016.
 908. Webster's ninth new collegiate dictionary, 1991.

909. B. F. Webster. *Pitfalls of object-oriented development: a guide for the wary and enthusiastic*. M&T books, New York, 1995.
910. L. Weinman and J. Lentz. *Deconstructing Web Graphics*. Macmillan, 1998.
911. M. V. Welie and G. C. V. der Veer. Pattern languages in interaction design. In *INTERACT*. IOS Press, 2003.
912. M. Weske. *Business Process Management: Concept, language, architecture*. Springer, 2007.
913. R. West and J. Leskovec. Human wayfinding in information networks. In *WWW*, pages 619–628, 2012.
914. B. Whorf. *Lost generation theories of mind, language, and religion*. Popular Culture Association, University Microfilms International, Ann Arbor, Mich., 1980.
915. J. Widom. Research problems in data warehousing. In *Proceedings of the 4th International Conference on Information and Knowledge Management*. ACM, 1995.
916. G. Wiederhold. *Database Design, Revised 2nd Edition*. McGraw-Hill, 2001.
917. R. Wieringa. *Requirements engineering - frameworks for understanding*. Wiley, 1996.
918. R. Wieringa. *Design methods for reactive systems: Yourdan, Statemate, and the UML*. Morgan Kaufmann, Amsterdam, 2003.
919. R. Wieringa. *Design science methodology*. Springer, Berlin, 2014.
920. E. Wilde. *Advanced XML Technologies*. CRC Press, 2004.
921. R. Wilhelm, H. Seidl, and S. Hack, editors. *Compiler Design: Syntactic and Semantic Analysis*. Springer, 2013.
922. A. S. Wilkins, editor. *M. Tulli Ciceronis Rhetorica. M. Tullius Cicero*. Clarendon Press Oxford Classical Texts, 1902.
923. M. Winckler and P. A. Palanque. StateWebCharts: A formal description technique dedicated to navigation modelling of web applications. In *Interactive Systems: Design, Specification, and Verification*, volume 2844 of *LNCS*, pages 61–76. Springer-Verlag, 2003.
924. G. Windsor. Data modeling tools. <http://www.databaseanswers.org/modelling-tools.htm>, accessed Sept. 28, 2016, 2015.
925. P. Winston. *Artificial Intelligence*. Pearson, 1992.
926. P. Wisse. *Metapattern - Context and time in information models*. Addison-Wesley, Boston, 2001.
927. S. Wolf. *Mensch - Maschine - Metapher. Zur Exemplifikation des menschlichen Geistes durch den Computer*. PhD thesis, Universität Bamberg, 1994.
928. U. Wolffgang. *Modellgetriebene Entwicklung daten- und prozessbasierter Webapplikationen*. PhD thesis, University of Münster, 2012.
929. W. A. Woods. What's in a link: Foundations for semantic networks. In *Representation and Understanding*, pages 35–82. Academic Press, New York, 1975.
930. C.-C. Yang. *Relational Databases*. Prentice-Hall, Englewood Cliffs, 1986.
931. Y. Yesilada, C. Jay, R. Stevens, and S. Harper. Validating the use and role of visual elements of web pages in navigation with an eye-tracking study. In *WWW*, pages 11–20, 2008.
932. S. Yigitbasi. Entwicklung und Einsatz einer Umwelt- und Forschungsdatenbank für die Lausitzer Bergbaufolgelandschaften. In *Kurzfassungen - 10. Workshop "Grundlagen von Datenbanken"*, *Workshop des GI-Arbeitskreises in Konstanz, 2.6.-5.6.1998*, volume 63 of *Konstanzer Schriften in Mathematik und Informatik*, pages 148–152. Universität Konstanz, 1998.

933. S. Yongchareon, C. Liu, X. Zhao, and J. Xu. An artifact-centric approach to generating web-based business process driven user interfaces. In *WISE*, pages 419–427, 2010.
934. R. Young. *The requirements engineering handbook*. Artech House, 2004.
935. E. S. K. Yu and J. Mylopoulos. Why goal-oriented requirements engineering. In *Proc. REFSQ*, pages 15–22. Presses Universitaires de Namur, 1998.
936. M. Yudelson, P. I. Pavlik, and K. R. Koedinger. User modeling - a notoriously black art. In *UMAP*, volume 6787 of *Lecture Notes in Computer Science*, pages 317–328, 2011.
937. G. Yule. *The study of language*. Cambridge University Press, Cambridge, 2003.
938. N. M. Yusoff and S. S. Salim. A review of storyboard tools, concepts and frameworks. In *Proc. Learning and Collaboration Technologies, HCI, Part I*, volume 8523 of *Lecture Notes in Computer Science*, pages 73–82. Springer, 2014.
939. J. A. Zachman. A framework for information systems architecture. *IBM Systems Journal*, 38(2/3):454–470, 1999.
940. S. Zahran. *Software Process Improvement. Practical Guidelines for Business Success*. Addison-Wesley, 1997.
941. K. Zettsu, K.-S. Kim, Y. Kidawara, and Y. Kiyoki. Towards moving phenomena data management. In *IUCS*, volume 398 of *ACM International Conference Proceeding Series*, pages 269–272. ACM, 2009.
942. T. Zeugmann. Inductive inference of optimal programs: A survey and open problems. In J. Dix, K. P. Jantke, and P. Schmidt, editors, *Nonmonotonic and Inductive Logics*, pages 208–222. Springer-Verlag, Berlin, 1991.
943. Y. Zhang. *Quality Modelling and Metrics of Web-based Information Systems*. PhD thesis, Oxford Brookes University, UK, 2005.
944. Y. Zhang, J. X. Yu, and J. Hou, editors. *Web Communities – Analysis and Construction*. Springer, 2006.
945. Y. Zhu and D. Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *VLDB*, pages 358–369. Morgan Kaufmann, 2002.
946. W. Zimmermann and B. Thalheim. Preface. In *ASM 2004*, number 3052 in *LNCS*, pages V–VII, Berlin, 2004. Springer.
947. O. Zschau, D. Traub, and R. Zahradka. *Web Content Management. Websites professionell planen und betreiben*. Galileo Business, Bonn, 2002.
948. S. Zurabyan. *Fundamentals of Bioorganic Chemistry*. Geotar-Media, Moscov, 2012.

Index

- A2C, *see* administration-to-consumers
Abstract State Machine, 393
abstraction layer model, 4, 20
 business layer, 20
 conceptual layer, 20
 data specification, 20
 dialogue specification, 21
 focus, 20
 function specification, 20
 implementation layer, 20
 modelling tasks, 21
 modus, 20
 presentation layer, 20
 strategic layer, 20
 view specification, 21
action scheme, *see* plot
activity demand, 398
actor
 information portfolio, 93
 obligation, 94
 rights, 94
 role, 70, 93
 user type, 70
actor context, 184
 actor ambiguity context, 185
 actor approximation context, 185
 actor mental context, 185
 actor projection context, 185
actor modelling, 91
actor portfolio, 158
 life case refinement, 162
actor profile, 156
 template, 157
adaptivity, *see* cohesion
 by aggregation, 289
administration-to-customers, 201
admitted story, 79
ALM, *see* abstraction layer model
ambience type, 317
application context, 7
application domain
 facets, 471
application domain description, 431
 concepts, 431
 stages, 431
application domain specification, 430
ASM method, *see* Abstract State
 Machine
aspects of WIS
 intention, 10
atmosphere category, 44
atmosphere of a WIS, 44
B2A, *see* business-to-administration
B2B, *see* business-to-business
B2C, *see* business-to-consumers
BPMN, 385
brainstorming, 22
 roles, 23
 tasks, 23
brand of a WIS, 33
 classification, 34
business culture, 210
business layer
 storyboard, 20
 storyboarding, 22, 61
 user profiling, 22

- business-to-administration, 201
- business-to-business, 13, 201
- business-to-consumers, 13, 201
- BV constraint, 382
- canonical order, 395
- Capability Maturity Model, *see* CMM
- Capability Maturity Model Integration, *see* CMMI
- cardinality constraint, 379
- Church-Rosser property, 127
- CIB, *see* contextual information base
- class-wise representation, 383
- CMM, 430
 - maturity level, 430
- CMMI, 429
- co-design
 - actor, 411
 - aspects, 445
 - communication act, 412
 - container, 421
 - content organisation, 418
 - content presentation, 418
 - databases, 377
 - distribution, 378
 - functionality, 378
 - infotainment, 412
 - interaction, 378
 - navigation, 419
 - plot, 411
 - retrieval, 419
 - scenario, 412
 - search, 419
 - SPICE conformity, 452, 453
 - stepwise development, 429
 - story, 411
 - story space, 411
 - structuring, 378
 - transformation, 418
 - web interaction type, 415
 - web page configuration, 421
 - web page extraction, 421
- co-design method, 471
 - activities, 474
 - application domain description, 472
 - architecture, 487
 - aspects, 476
 - blueprint, 492
 - brainstorming, 478
- extensions, 489
- partners, 477
- production, 518
- quality, 524
- requirements, 472
- requirements analysis, 497
- resources, 476
- results, 476
- separation of concerns, 490
- social spheres, 516
- steps, 474
- storybook, 505
- tips and tricks, 478
- view development, 521
- work products, 472
- workflow, 520
- cohesion
 - adaptation of operations, 288
 - cohesion preorder, 285
 - proximity values, 287
- collaboration, 162
- collaboration pattern, 408
- colour chord, 317
- colour ordering, 317
- colour schema
 - colour chord, 319
- colour scheme
 - effect, 318
- commercial-off-the-shelf, *see* COTS
- communication
 - localisation abstraction, 52
 - metaphorical structure, 53
- communication act, 137, 412
 - dimension, 137
 - saga, 137
- communication analysis, 50
- communication flaw, 50
- communication matrix, 84
- community system
 - actors, 214
 - branding, 213
 - content, 16
 - content chunks, 221
 - functionality, 17, 219
 - portfolios, 217
 - purpose, 11
 - scenarios, 219
 - user profiles, 14
 - verb fields, 216

- conceptual layer
 - adaptivity, 25
 - content, 24
 - content type, 25
 - database, 24
 - functionality, 24
 - scenario support, 25
 - transaction, 25
 - view, 25
 - web interaction schema, 25
 - web interaction type, 24
- conceptual modelling, 24
- conceptual structures
 - storyboarding, 135
- confluent, 127
- container, 329
 - building blocks, 331
- container specification, 331
- content aspect
 - conceptual level, 15
 - logical level, 15
- content chunk, 170
 - entry scene, 177
 - template, 177
- content context space, 191
- content data type, 263
- content relationship, 38
- content schema, 263, 358
 - integrated, 359
- content-extended scenario, 174
- context, 6
 - application, 7
 - template, 189
 - user, 7
 - WIS, 7
- context layer, 43
- context manager, 192
- context space, 7, 181
- context theory
 - adaptivity, 192
 - lifting relation, 190
- context type, 309
- contextual information base, 304
 - context, 305
- coordination
 - contract, 409
- COTS
 - WIS engineering, 449
- cube
 - cell, 396
 - dimension, 396
 - fact, 396
 - grounding schema, 396
 - integration, 397
 - named, 396
- cultural factors, 346
- cultural model
 - Hofstede model, 339
 - Lewis model, 339
- cultural stereotype
 - in interface design, 351
- culture
 - adaptation, 349
 - aspects, 338
 - components, 337
 - definition, 335
 - layered structure, 336
 - learning aspect, 337
 - linear, 356
 - multi-active, 356
 - onion model, 336
 - persona, 346
 - reactive, 356
 - stereotype, 338
 - user portfolio, 347
 - user profile, 347
- data consumption, 71
- data cube, 396
- data production, 71
- data type, 262
 - hierarchical, 395
- database farm, 410
- database operation, 391
- database schema
 - association of cultural stereotypes, 359
 - parametric, 357
 - stereotypes, 357
- database type, 262
- developer context, 184
- dialogue, 413
 - postcondition, 414
 - precondition, 414
- dialogue box, 275
- disc function, 397
- distributed system
 - collaboration, 407

- communication, 407
- cooperation, 408
- coordination, 407
- distribution
 - architecture, 408
 - exchange frame, 410
- docket, 422
- drill-down function, 397
- e-business
 - actors, 203
 - branding, 202
 - content, 15
 - portfolio scoping, 206
 - production rules, 207
 - purpose, 11
 - scenario, 66
 - scenarios, 204
 - stories, 14
 - support features, 209
 - user profile, 13
 - utilisation portfolio, 208
 - verb fields, 204
 - WIS portfolio, 211
- e-commerce
 - functionality, 17
 - metaphor, 11
 - purpose, 11
- e-learning
 - authentication, 14
 - content, 16
 - functionality, 18
 - metaphor, 12
 - purpose, 12
 - scenario, 67
 - usage, 14
- education profile, 152
- edutainment
 - authentication, 14
 - content, 16
 - purpose, 12
 - usage, 14
- edutainment system, *see* learning WIS
- entertainment
 - functionality, 17
- entertainment system, 222
 - intention, 224
 - purpose, 12
 - usage, 14
- entity-relationship model, 379
- entry scene
 - template, 179
- episode, *see* scenario
- equality-generating constraint, 383
- ETL, 361
- extract-transform-load, *see* ETL
- facets of context, 182
- facets of intention, 138
 - aims, 138
 - intent facet, 138
 - object, 139
 - objectives, 138
 - purpose, 138
 - representation facet, 139
 - targets, 139
 - time facet, 139
- Fibonacci grid, 320
- Fibonacci sequence, 320
- Fibonacci spiral, 320
- forum community, 218
- functionality
 - navigation, 17
 - storyboard, 17
- functionality chunk, 170, 180
- general join, 265
- golden section, 320
- greatest consistent specialisation, 385
- group system, *see* community system,
 - see* community system
- HCI, 313
- HERM, 380
 - attribute, 380
 - cardinality constraint, 383
 - cluster type, 380
 - database, 269
 - database schema, 269
 - database type, 268
 - entity type, 380
 - integrity constraint, 380, 382
 - relationship type, 380
 - representation of XML, 416
 - view, 388
- HERM diagram, 380
- hierarchy of scenes, 71
- higher-order Entity-Relationship model,
 - see* HERM

- Hofstede model
 - dimensions, 339
- human computer interaction, *see* HCI
- human-computer system
 - dochotomy, 400
- identity WIS
 - adaptation, 234
 - branding, 225
 - communication profile, 228
 - content, 16
 - corporate identity, 226
 - functionality, 18
 - information portfolio, 232
 - purpose, 12
 - scenarios, 231
 - specification of identity, 229
 - storyboarding, 14
 - user profiling, 14
 - verb fields, 233
 - word fields, 231
- implementation layer
 - database implementation, 26
 - operation implementation, 26
 - page generation, 26
 - view implementation, 26
 - XSL, 26
- individual culture, 338
- information
 - anthropomorphic definition, 398
- information consumption, 22
- information demand, 170, 361, 398
- information logistics, 363
- information need, 170, 361
- information portfolio, 170
- information production, 22
- information search, 363
 - dependence on culture, 366
 - query-answer forms, 365
 - questions, 364
 - self-organisation, 364
- information service
 - actors, 245
 - content, 16
 - content chunks, 247
 - functionality, 18, 252
 - life cases, 250
 - metaphor, 12
 - purpose, 12
- quality assurance, 253
- scenario specification, 245
- story space, 247
- system organisation, 249
- user profiles, 15
- information system
 - profile, 367
 - service support, 406
 - system perspective, 399
 - user perspective, 399
- information transfer, 136
- informative model, 367
 - cargo, 367
 - system profiling, 368
- infotainment, *see* information service
 - HERM schema, 380
- input form, 369
 - defaults, 369
- integration of scenarios, 64
- integrity constraint, 379
 - implicit, 382
 - model-inherent, 382
- integrity enforcement, 385
- intention
 - template, 144
- intention aspect, 10
 - metaphor, 11
 - time scale, 11
- interaction object, 262, 399
- interaction schema, 262
- interaction type, 262, 263, 399
 - adaptivity extension, 291
 - aggregation operation, 278
 - browsing operation, 278
 - contextual extension, 305
 - extension by hierarchical versions, 294
 - generalisation operation, 278
 - generic functionality, 277
 - hierarchical versions, 292
 - join operation, 279
 - link operation, 279
 - logic program, 271
 - measuring extension, 300
 - operation, 276
 - ordering extension, 301
 - presentation extension, 302
 - presentation operation, 280
 - query algebra, 264

- reordering operation, 278
- search operation, 279
- sequentialisation operation, 279
- specialisation operation, 278
- visualisation operation, 280
- interface
 - guidelines, 349
- ISO/IEC 15504, 436
- interaction object, 263
- join function, 397
- KAT, *see* Kleene algebra with tests
 - term rewriting, 118
- Kiviat graph, 100
- Kleene algebra, 112
- Kleene algebra with tests, 113
- learning WIS
 - branding, 236
 - classification, 243
 - content, 238
 - scenarios, 237
 - support features, 242
 - verb fields, 240
 - word fields, 237
- Leonardo da Vinci, 320
- Lewis model
 - factors, 340
- life case, 144
 - concept, 146
 - development, 147
 - extension, 191
 - refinement, 167
 - specialisation, 191
 - template, 150
 - template extension, 168
- life case analysis, 148
- life case detection
 - interview techniques, 149
- lifespan pattern, 329
- lifting relation, 190
- linguistic analysis, 9, 46
- linguistic reflexivity, 386
- local as view, 379
- locally confluent, 127
- majority order, 395
- measuring system, 300
- media type, *see* web interaction type
- mediation service, 399
- metaphor, 22
 - storyboarding, 195
- metaphorical structure, 194
- mini-story, 354
 - adaptation to culture, 355
 - parametric, 354
 - refinement, 372
- mission statement, 10, 33
 - description, 35
 - metaphor, 11, 35
 - semi-formal text, 36
- MOCCA, 352
- model, 4
 - abstraction property, 4
 - mapping property, 4
 - pragmatics, 4
 - reduction property, 4
- modelling, 4
 - pragmatics, 4
 - semantics, 4
 - syntax, 4
- named cube, 396
- national culture, 337
- object-generating constraint, 383
- object-wise representation, 384
- onion approach, 415
- operation
 - on a database type, 273
 - on an interaction type, 276
 - selection type, 275
- organisational context, 184
- organisational culture, 338
- output form, 370
- parametric action, 355
- persona
 - concept, 172
 - specification template, 173
- personalisation, *see* plot
- personality profile, 154
- pivoting function, 397
- placement grid, 332
- play-out system, 384
- plot, 75
 - compatibility, 80
 - customisation to preferences, 116

- personalisation, 116
- polarity profile, 155
- portfolio context, 166
- pre-site, 263
- preference condition, 115
- preference rule, 101
 - deontic constraint, 128
- presentation layer
 - style option, 26
- presentation rule, 302
- principle of six big W, 5
- professional culture, 338
- programming in the web, 407
- progression pattern, 44
- project culture, 338
- provider context, 184
- purpose of a WIS, 10
- QuASAR approach, 493
- query
 - logic program, 271
- rational tree, 266
- rational tree type, 267
- refinement, 371
- renaming function, 397
- requirements elicitation, 433
- rewrite rule, 119
- role
 - deontic logic, 94
- roll-up function, 397
- rotation function, 397
- scenario, 22, 62
 - acceptance condition, 69
 - action, 22
 - content-extended, 174
 - details, 68
 - enabling event, 69
 - graph, 64
 - postcondition, 68
 - precondition, 68
 - triggering event, 69
- scene, 62
 - actor, 70
 - grouping arrangement, 310
- screen tiling, 319
 - rhythmic structure, 320
- screenography, 313
- ambience type, 317
- architecture, 314
- atmosphere, 318
- cognitive aspects, 320
- guidelines, 325
- intention, 315
- layout, 314
- layout patterns, 319
- life cases, 316
- playout, 314
- tiling, 319
- user models, 315
- utilisation portfolio, 328
- utilisation space, 328
- search
 - culture awareness, 359
 - kinds, 360
- selection type, 275
- semantic tree, 38
- semiotics, 134
 - pragmatics, 134
 - semantics, 134
 - syntactics, 134
- semiotics triangle, 134
- service architecture, 398
- service specification, 390
- service system, 404
- session, 308
- session type, 308
- site, 295, 306
- site culture, 338
- SiteLang, 76
- SiteLang editor, 327
- situational culture, 338
- six big W, 5
- slice function
 - basic, 397
 - close, 397
 - eager, 397
 - lazy, 397
 - liberal, 397
- snowflake schema, 395
- social context, *see* organisational context
- socio-technical system, 398
 - Janus character, 398
- software development process
 - quality, 429
- SPICE, 429

- capability level, 451
- in WIS engineering, 451
- levels, 430
- processes, 451
- spreadsheet cube, 396
- star schema, 395
- statechart, 89
- status set, 129
 - enabled sequence, 129
 - feasible, 129
- step algebra, 385
- stored procedure, 385
- story, 13, 22
 - tuning, 24
- story portfolio, 179
- story space, 62
 - plot, 76
- storybook, 24, 106
 - culture awareness, 354
 - description catalogue, 87
 - enhancement, 371
 - KAT, 113
 - scene, 62
- Storyboard context, 187
 - post-scene context, 187
 - pre-scene context, 187
 - scene context, 188
- storyboarding, 13, 61
 - context, 23
 - scenario, 24
 - scene, 23
- strategic layer
 - ambiente, 34
 - atmosphere, 34
 - brainstorming, 22
 - branding, 33
 - guidelines, 33
 - intentions, 21
 - modelling, 33
 - user roles, 21
 - user tasks, 21
 - WIS presentation, 33
- structural recursion, 264
- structure expression, 263
- structure pattern, 357
- subtask, 103
- subtyping, 265
- system architecture, 457
- system context, 188
- system ladder, 400
 - refinement, 400
- task, 103
 - role, 104
 - scene, 104
- task context, 159
- task execution model, 160
- task modelling, 103
- task specification, 405
- task-goal graph, 41
- team culture, 338
- temporal context, 188
- temporal database logic, 385
- transaction community, 217
- transformation rule, 393
- transition constraint, 385
- transition system, 384
- trigger, 385
- triptych paradigm, 430
- type system, 262
- union function, 397
- usage context, 180
- user
 - obligations, 22
 - rights, 22
- user context, 7
- user interaction
 - guideline, 348
- user model, 7
- user modelling
 - portfolio, 152
 - profile, 152
- user portfolio, 365, 404
- user profile, 13, 22, 23, 96, 365, 404
 - dimension, 97
 - template, 156
- user profiling, 13
- user role, 22
- user task, 22
- user type, 70, 99
 - Kiviat graph, 100
- utilisation context, 34, 42
 - metaphor, 43
- utilisation portfolio, 33
 - actor, 40
- utilisation space, 33
 - content, 37

- description, 37
- functionality, 37
- semantic tree, 38
- value representability, 382
- view
 - architecture, 387
 - parametric, 357
 - service support, 388
- view schema, 388
- view suite, 388
- view tower, 387
- visual cognition
 - principles, 322
- visual communication
 - cognition, 321
 - memorising, 321
 - principles, 321
 - processing, 321
 - vision, 321
- visual design
 - principles, 323
- visual elements, 323
- weak value representability, 382
- Web 2.0, 402
- Web 3.0, 402
- web information system, 3, 10
 - categories, 8
 - collaboration, 310
 - communities, 9
 - conceptual facilities, 401
 - conceptual structure, 402
 - concerns, 401
 - content, 15, 259
 - context, 18
 - design, 379
 - e-business, 8
 - e-learning, 8
 - edutainment, 9
 - entertainment, 9
 - form-based interface, 280
 - functionality, 16, 259
 - identity, 9
 - information service, 8
 - pattern, 8
 - presentation, 19
 - requirements, 7
 - six big W, 5
- web interaction object, 26, 295, 306
- web interaction schema, 294, 306
- web interaction type, 24, 294, 305
 - communication, 310
 - cooperation, 310
 - coordination, 311
- web page
 - design framework, 327
- web page design
 - third generation, 327
- web page grid, 327, 329
- web page pattern, 327
- web page suite, 327
- web site, 10
- web utilisation space, 135
- website context, 180
- WIS, *see* web information system
 - content aspect, 15
 - context, 7
 - context aspect, 18
 - functionality aspect, 16
 - information system, 435
 - presentation aspect, 19
 - presentation system, 435
 - quality, 466
 - realisation, 529
 - requirements capture, 432
 - specification, 435
- WIS context, 7
- WIS development
 - primary dimensions, 465
 - secondary dimensions, 465
 - semiotic background, 468
- WIS engineering
 - CottbusNet*, 463
 - activities, 437
 - activity cycle, 439
 - activity dimension, 437
 - architectural styles, 459
 - architectural views, 459
 - architecture, 457
 - architecture blueprint, 463
 - architecture-driven, 460
 - co-design, 445
 - COTS, 449
 - development aspects, 443
 - dialogue, 448
 - dimensions, 436
 - elicitation techniques, 434

- orchestration, 452
 - partner dimension, 442
 - pattern-based, 463
 - product dimension, 440
 - quality control, 451
 - resource dimension, 445
 - SPICE, 451
 - stakeholders, 443
 - steps, 438
 - turnkey development, 449
 - work product, 441
 - work products, 447
- WIS pattern, 8
- WIS portfolio, 169
- word field, 9, 46
- work profile, 153