
Requirements Engineering: Artefact-Based Customisation

Daniel Méndez Fernández



Technische Universität München

Institut für Informatik
der Technischen Universität München

**Requirements Engineering:
Artefact-Based Customisation**

Daniel Méndez Fernández

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. Florian Matthes

Prüfer der Dissertation:

1. Univ.-Prof. Dr. Dr. h.c. Manfred Broy
2. Univ.-Prof. Dr. Alexander Knapp,
Universität Augsburg

Die Dissertation wurde am 07.07.2011 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 20.10.2011 angenommen.

Abstract

Requirements engineering (RE) constitutes an important success factor for software development projects, since unambiguous and stakeholder-appropriate requirements are important determinants of quality. At the same time, RE is an interdisciplinary area in a software development process, which is driven by uncertainty. The necessity of integrating isolated RE techniques into company-specific development process models and the various influences that engineers have to face in volatile project environments increase the demand for an RE approach that

- captures the basic (modelling) concepts of an application domain in a reference model, serving in individual project environments as orientation to create syntactically complete and consistent RE results.
- gives guidance to
 - deeply integrate the reference model into a company-specific development process model.
 - effectively cope with uncertain project situations when applying the reference model at project level to create the RE results in direct response to individual, volatile project characteristics.

A promising solution to this problem is given by the artefact-based philosophy. In contrast to the activity-based philosophy, we would define an RE reference model of all artefacts that are an intermediate result of the RE process and that capture the basic (modelling) concepts of an application domain. Since the complexity of potentially differing processes is then reduced to the dependencies between the artefacts, we are able to systematically integrate the reference model into a development process model and to flexibly apply it at project level.

We take this idea and the knowledge we already gained from artefact-based approaches in the area of, e.g., development process models, like the V-Modell XT, as a motivation for the contribution of the thesis. We contribute in response to the problems stated above an *artefact-based customisation approach for requirements engineering*, which yet is missing.

Since this thesis aims, inter alia, at extending and integrating available approaches on the basis of an artefact-based philosophy, we first analyse possible notions of artefact orientation and infer a *meta model for artefact orientation* according to our particular research objectives. According to this meta model, we then contribute

- an *RE reference model for business information systems* as a domain-specific interpretation of the meta model. We develop and evaluate this reference model in an industrial setting to additionally ensure the validity for the application domain.
- an *artefact-based customisation approach* that
 - guides the process integration of the RE reference model into a development process model.
 - assists the customisation of the integrated RE reference model at project level in response to project characteristics. This customisation approach transfers related work, e.g., in the activity-based area of situational method engineering and decision support systems, to the artefact-based philosophy with respect to our meta model.

We finally contribute two case studies in order to evaluate our contributions in a qualitative manner with respect to our research objectives. We will show that our artefact-based customisation approach tackles the shortcomings in given (activity-based) approaches. This provides a first empirical basis on the advantages of artefact-based customisation for RE and closes an existing gap in literature.

Acknowledgements

I would like to thank all the people who supported me in the last years making this thesis possible. First of all, I want to thank my supervisor Prof. Dr. Dr. h.c. Manfred Broy, who was always very supportive in any imaginable way. I am very grateful for his confidence in me, for the possibility of working on this variety of challenging research projects, and for his faithful consults and guidance I got whenever I needed it. I also would like to thank my co-supervisor Prof. Dr. Alexander Knapp. He was always available for stimulating discussions and gave me valuable feedback on previous versions of this thesis.

Furthermore, I thank all my colleagues and friends of the research group *Software & Systems Engineering*, in particular:

- Dr. Birgit Penzenstadler, for continuously proof reading my thesis, for continuously calming me down every single minute I got worried about the thesis (which was rather often), and for continuously supporting me over two years with my research projects so that I could focus on my thesis. I want to thank her for her unconditional patience and for being a great friend.
- Dr. Marco Kuhrmann, for helping me to get the big picture of the world around requirements engineering, for fruitful discussions about the principles of modelling and meta-modelling, and for supporting me in the area of customisation of development process models. I guess our friendship has grown strong enough to even survive his incredibly bad jokes.
- Prof. Dr. Stefan Wagner, for teaching me the essence of good empirical research and scientific writing and for teaching me how to correctly approach problems in a scientific manner; for presenting a paper at a conference I couldn't attend on short notice, and for always being willing to provide me with fruitful discussions, even when his calendar was about to burst.
- Silke Müller, Marina Franke, Eleni Weiss, and Andrea Heller, for their great administrative work at the research group.
- Franz Huber, Dieter Mletzko, and the MTA team, for providing us with an IT infrastructure of which many others could learn.

I am also very grateful to the employees at Capgemini Technology Services that participated in the field study on requirements engineering and in the development of Quasar Requirements. The work with them has set the context for working on this thesis. In particular, I thank Prof. Dr. Andrea Baumann, for encouraging me, together with Oliver Flöricke, throughout the whole research co-operation so that I was able to focus on my topic. I also thank the persons who provided me with discussions related to my research, in particular, Dr. Gerhard Koller, Dr. Frank Salger, and Prof. Dr. André Schekelmann, as well as Dr. Stefan Sauer, Prof. Dr. Gregor Engels, and Michael Mlynarski of the University of Paderborn. I am also very grateful to the employees of Siemens that participated in the case study.

Furthermore, I would like to thank Christoph Moosbauer. He initially encouraged me to apply for a position at the university. Lacking his advice, I would have probably ended up with a much more boring job and a ridiculously high salary. In fact, he was the one who helped me to make the best decision I could ever make when I most needed a good advice.

Finally, I want to thank my parents and my brother Javi. They were, and are, always willing to support me in any possible way. Acknowledging the rest of my family would result in a whole book. However, I want you to know that my thoughts are always with all of you being spread all over the world. Over and above all, Sara-Rose: I am deeply grateful to you for unconditionally standing by my side for the last 10 years and for accepting my perfectly planned, although not-so-perfectly performed proposal.

Contents

1	Introduction	1
1.1	Problem Statement	2
1.2	Research Objectives	5
1.3	Related Work	6
1.3.1	Activity Orientation	6
1.3.2	Artefact Orientation	7
1.4	Contribution	8
1.5	Research Method	12
1.6	Outline of the Thesis	14
1.7	Previously Published Material	14
2	Fundamentals and Related Work	15
2.1	Outline of Fundamentals and Related Work	16
2.2	Requirements Engineering and Management	17
2.2.1	Discipline and Objectives	17
2.2.2	Requirements and their Classification	19
2.2.3	Refinement Interdependencies and Traceability	21
2.2.4	RE in Context of the Development Life Cycle	24
2.2.5	Overview of Possibilities for Tool Support	25
2.3	Fundamentals in Business Information Systems	26
2.3.1	Principles and Architecture Reference Models	26
2.3.2	Architecture Frameworks for Business Information Systems	31
2.4	Requirements Engineering for Business Information Systems	32
2.5	Development Process Models	34
2.5.1	Principles and Terms in Development Process Models	34
2.5.2	Setting of Contributions in Context of Development Process Models	37
2.5.3	Related Work in Development Process Meta Models	39
2.6	Customisation of Development Process Models	45
2.6.1	Customisation Stages and Relation to Thesis	45
2.6.2	Classification of Related Work	48
2.6.3	Domain- and Discipline-specific Contributions	50
2.6.4	Contributions in Development Process Models	52
2.6.5	Summary and Discussion of Related Work	56
2.7	Introduction of a Running Example	57

3	Artefact Orientation	59
3.1	Principles of Artefact Orientation	60
3.1.1	Artefact Orientation versus Activity Orientation	60
3.1.2	Potential Objectives in Artefact Orientation	64
3.1.3	Potential Variations of Artefact Models	65
3.1.4	Variations of Artefact Models according to Objectives	71
3.2	Design of a Meta Model for Artefact Orientation	73
3.2.1	Research Objectives and Requirements for Artefact Orientation	73
3.2.2	Design of a Meta Model for Artefact Orientation according to Research Objectives	75
3.3	Meta Model for Artefact Orientation	77
3.3.1	Overview of the Meta Model	77
3.3.2	Sub-Models of the Meta Model	78
3.3.3	Validity Procedure during Instantiation of the Meta Model . .	83
4	Artefact-based Reference Model	89
4.1	Business Information Systems Analysis	90
4.2	Artefact Abstraction Model	91
4.2.1	Levels of Abstraction and Modelling Views	92
4.2.2	Basic Concepts and their Decomposition and Refinement . . .	95
4.2.3	Solution-Oriented and Problem-Oriented Refinement	101
4.3	Artefact Model Overview	105
4.3.1	Overview of Artefact Types	105
4.3.2	Representation of Artefact Types in the Thesis	105
4.3.3	Overview of Dependencies in the Artefact Model	107
4.3.4	Artefact Status Model for Progress Control	110
4.4	Artefact Type: Business Specification	111
4.4.1	Structure Model	111
4.4.2	Content Model	115
4.4.3	Syntax	124
4.5	Artefact Type: Requirements Specification	126
4.5.1	Structure Model	126
4.5.2	Content Model	129
4.5.3	Syntax	139
4.6	Artefact Type: Traceability Matrix	141
4.7	Artefacts as Interfaces for a Process Integration	142
4.8	Process Model	143
4.8.1	Overview of Process Model	144
4.8.2	Properties supporting Customisation at Project Level	144
4.8.3	Milestones	146
4.8.4	Activities for Content Creation	146
4.9	Role Model	147
5	Artefact-Based Customisation	151
5.1	Customisation Overview	152
5.2	Process Integration	153
5.2.1	Recapitulation of Related Work	153
5.2.2	Approach for a Process Integration	154
5.3	Customisation at Project Level	158
5.3.1	Recapitulation of Related Work and Transfer	159
5.3.2	Principles of Artefact-based Customisation	160
5.3.3	Approach for an Artefact-based Customisation	167
6	Case Studies	173

6.1	Case Studies Overview	174
6.2	Case Study 1: Process Integration	175
6.2.1	Research Objective and Context	175
6.2.2	Case Study Design	177
6.2.3	Case Study Results	181
6.3	Case Study 2: Customisation at Project Level	189
6.3.1	Research Objective and Context	189
6.3.2	Case Study Design	190
6.3.3	Case Study Results	195
6.4	Discussion and Conclusion	201
6.4.1	Summary of Conclusions	201
6.4.2	Impact / Implications	203
6.4.3	Relation to Existing Evidence	203
6.4.4	Case Study Limitations	203
7	Summary and Outlook	205
7.1	Summary of Contributions	206
7.2	Future Work	209
A	Glossary of Terms used in the Thesis	213
B	Artefact Model	221
B.1	Structure Model and Possibilities for Arrangement	221
B.1.1	Structuring of the Business Specification	221
B.1.2	Structuring of the Requirements Specification	222
B.2	Comprehensive Content Model	225
B.2.1	Concepts of the Business Specification	225
B.2.2	Concepts of the Requirements Specification	226
B.2.3	Dependencies between Concepts of the Business and the Re- quirements Specification	227
B.2.4	Dependencies between Concepts of the Requirements and a System Specification	227
C	Case Studies: Additional Background Information	231
	Bibliography	235

List of Figures

1.1	Field study overview.	2
1.2	Overview of the contributions of the thesis.	9
1.3	Research method chosen in the thesis.	12
2.1	Outline of related work and relation to contributions.	16
2.2	Requirements taxonomy by Glinz [Gli07].	20
2.3	Interdependency types in the context of the thesis.	23
2.4	Areas of tool-support.	25
2.5	Levels of abstraction in the capgemini IAF [Gro06a].	32
2.6	Extent of requirements engineering in the Capgemini IAF.	33
2.7	Sub-models of development process models.	36
2.8	Packages of the <i>Software Process Engineering Meta-Model</i> (SPEM).	41
2.9	Packages of the V-Modell XT meta model defined in [TK09].	42
2.10	Static view on product types.	43
2.11	Product type dependencies.	44
2.12	Associations of product types to further elements.	44
2.13	Classification of related work.	50
3.1	Comparison of philosophies (simplified).	61
3.2	Variations of artefact models.	66
3.3	Variations of artefact models with characteristics.	71
3.4	Sketch of meta model for artefact orientation according to requirements.	76
3.5	Meta model for artefact orientation.	78
3.6	Development procedure of the artefact-based reference model.	84
4.1	Scope of <i>Business Information Systems Analysis</i> (BISA).	90
4.2	Levels of abstraction and modelling views in BISA.	92
4.3	Concept types in relation to artefact abstraction model.	96
4.4	Quality-related concepts in relation to artefact abstraction model.	100
4.5	Problem-oriented refinement versus solution orientation.	104
4.6	Overview of artefact types and (top-level) content items.	106
4.7	Overview of (simplified) dependencies.	107
4.8	Artefact status model.	110
4.9	Structure of the business specification.	112
4.10	Concepts for modelling structure.	115
4.11	Concepts for modelling (business) behaviour.	117

4.12	Concepts for modelling data.	120
4.13	Concepts for modelling business goals and business roles.	122
4.14	Structure of the requirements specification.	127
4.15	Basic requirements concept types and attributes.	130
4.16	Concepts for modelling system behaviour.	132
4.17	Concepts for modelling data.	136
4.18	Concepts for modelling actors.	137
4.19	Concepts for modelling system quality requirements.	138
4.20	Overview of Business Information Systems Analysis.	144
4.21	Iterative content creation.	145
4.22	Milestones in relation to artefact model.	146
5.1	Overview of artefact-based customisation approach.	152
5.2	Overview of approach for a process integration.	154
5.3	Order in which to integrate the elements of the meta model.	156
5.4	Variations for customising a development process model.	157
5.5	Overview of customisation approach at project level.	160
5.6	Project characterisation following the artefact-based philosophy.	161
5.7	Variations in a project execution and decision gates.	163
5.8	Artefact-based decision support.	165
5.9	Structure of the BISA diary.	167
5.10	Approach for an initial project set-up.	168
5.11	Approach for a project-specific BISA execution strategy.	170
6.1	Overview of case studies.	174
6.2	Overview of data collection procedure.	178
6.3	Sketch of the process integration with particular focus on artefacts (simplified).	183
6.4	Creation of disciplines and corresponding product types.	184
6.5	Creation of procedure modules for project execution strategies.	184
6.6	Analysis results shown as radar plots.	186
6.7	Overview of data collection procedure.	192
6.8	Exemplary (operations) use cases.	196
6.9	Analysis results shown as radar plots.	198
B.1	Structuring the business specification.	222
B.2	Structuring the requirements specification (1/2).	223
B.3	Structuring the requirements specification (2/2).	224
B.4	Concept model of the business specification.	225
B.5	Concept model of the requirements specification.	226
B.6	Dependencies between concepts of the business and the requirements specification.	227
B.7	Dependencies between requirements concepts and system concepts.	228
C.1	Template used during elicitation workshops.	231
C.2	Screenshot: Enterprise Architect extensions.	232

CHAPTER 1

Introduction

A large chunk of the Computer Science community do not think that it is an engineering discipline, and hence do not understand what's involved in working on real problems.

Ian Sommerville

Requirements engineering (RE) lays the foundation for successful software and system development projects regarding cost and quality [Bro06]¹. The activities, which are performed as part of the RE process, aim at the specification of requirements that unambiguously reflect the purpose of what a software system is intended for, as well as the needs of all relevant stakeholders [FGP04, Gei05]. The precise definition of requirements supports subsequent software development activities like architectural design, (acceptance) testing, or project management. Requirements are used, for example, as a basis for the definition of contractual agreements [HWFP07]. As a software engineering discipline, RE contributes with precise requirements specifications to appropriateness and cost-effectiveness in the development of a system [NE00]. Hence, RE provides the basis for efficient and effective, customer-oriented development projects and, thus, is an important success factor for productivity and (product) quality [DC06].

The first step for companies towards RE excellence is the definition of an RE process at *organisational level* (see also Fig. 1.1 on page 2). This includes the preparation of methods and techniques for a company-wide use to enable knowledge transfer among different projects [Bri96, NSK00]. Furthermore, the RE process must be integrated into the development life cycle. This integration concerns establishing the associations of RE-specific results with further contents of a development process model and is known as *process integration*²; for instance, by establishing the associations of use cases, which exemplarily describe how users intend to interact with a system, with project management activities that might perform on the basis of those use cases estimations of complexity and effort.

¹ In the context of the introduction we use the term *requirements engineering* in the broader sense. In Sect. 2.3, we give an overview of the application domain of business information systems and infer in Chp. 4.1 a domain-specific interpretation of RE covering aspects of requirements engineering and aspects of business engineering.

² In Sect. 2.5.1, we discuss relevant terms in the area of development process models.

1.1 Problem Statement

Once the RE process has been defined and integrated into the development process of a company, it has to guide the systematic creation of requirements specifications at *project level*. This considers each individual project of a company in which the participants act according to the integrated RE reference model (def. in Sect. 2.5.1.1). The RE process that is performed at project level, must support the creation of precise and reproducible requirements specifications. *Preciseness* in the creation of requirements specifications includes aspects of structure, syntax, and semantics regarding the content of the specification documents. Obviously, the (reference) process at organisational level must include the basic concepts of an application domain. The term “application domain” considers the characterisation of an area (environment) in which a particular family of systems (e.g., business information systems) is used.

Having an RE approach that captures the concepts of an application domain achieves for requirements engineers *awareness* of creating requirements specifications that are *conformant* to the domain-specific reference model with respect to *completeness* and *consistency* in the results. In our context, completeness considers the creation of the results in a syntactically complete manner with all necessary elements, e.g., that each use case has a unique identifier. Consistency considers the syntactic consistency in the relations between the results; for instance, that each use case description needs the explicit and separate description of an actor. Both completeness and consistency in the results support *seamless modelling* and, thus, *continuity* in the process.

Outline of the Chapter. We now analyse a major observable problem, which consists of a missing integration of RE at organisational level and that extends, at project level, to the missing support of a systematic RE. We discuss reasons and resulting phenomena of solution orientation in RE affecting the preciseness of produced requirements specifications. We formulate our research objectives and discuss related work. We define the contributions of the thesis that tackle the shortcomings of related work, give an overview over the chosen research method, and conclude the chapter with an outline of the thesis and a summary of previously published material. Please refer also to the glossary of terms in appendix A, since we already use terms that are introduced and defined in subsequent chapters.

1.1 Problem Statement

We performed a field study on RE [MFWLB10, MFWL⁺12] and investigated two major RE aspects at the company Capgemini Technology Services (see Fig. 1.1).

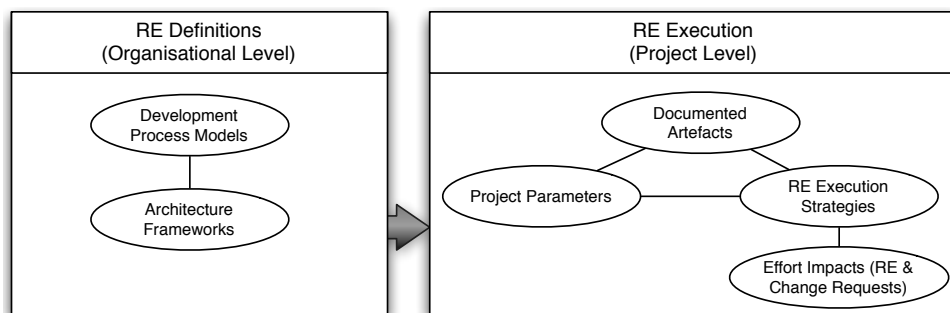


Figure 1.1: Overview of factors analysed in our field study on RE [MFWLB10, MFWL⁺12].

First, we investigated the definition of RE at an organisational level considering the company-wide development process model and domain-specific architecture frameworks used. Second, we investigated a set of projects with respect to the execution of their RE processes. To this end, we investigated which artefacts were produced, which project parameters (characteristics) related to the creation of those artefacts, a set of RE execution strategies, which show similarities among the quality of artefacts created in different projects in response to project parameters; and the effort spent on the artefacts' creation, including potentially resulting change requests used to discuss the efficiency of the execution strategies. This made possible the discovery and evaluation of current problems and needs in RE and allowed for the inference of an appropriate contribution to tackle the identified problems.

In the following, we discuss the study results and reasons that led to those results, before concluding our problem statement with a summary. Detailed information about the study design and the results can be taken from our contributions in [MFWLB10, MFWL⁺12]. Information about the motivation for the study and further relations of the study results to our contributions can be taken from our research method in Sect. 1.5.

Missing Integration of RE at Organisational Level. One observation made during the study was the missing integration of RE at organisational level. The definition of a (micro) process and its integration into a development process generally depends on the application domain. The application domain, reflected in (architecture) reference models and principles, impacts, in turn, on the used methods and the description techniques, which capture basic modelling concepts. There exists for each application domain a variety of approaches and methods that differently affect the understanding of RE and corresponding development process models.

Regarding business information systems, existing architecture frameworks arise from decades in which RE was neglected to be an integrated part of the software development life cycle [Boe06]. We can see the outcome in historically grown frameworks, such as in the *Zachmann framework* [Zac87], or (as it is the case in the field study) in the *Integrated Architecture Framework* (IAF) [Gro06a], which is an industrial derivative of the Zachmann framework developed by the Capgemini Group. An introduction to the fundamentals of the envisioned application domain and a description of exemplary enterprise architecture frameworks is given in Sect. 2.3. Even if the philosophy of "IT Business Alignment" considers principles of RE, mentioned frameworks still do not integrate the artefacts of RE. Existing frameworks are dominated by architectural solution crafting in which requirements are narrowed down to be a prerequisite for most of the defined results (see also [Sch04]). As a consequence of these prejudices, observable misunderstandings and unawareness of requirements and requirements specifications reduce these results to be the main input for business engineering activities, instead of being a core result of them (as seen by other approaches such as [Wie03, DKOA06, BCVP06]). Hence, on the one hand, RE is recognised to be part of engineering activities for constraining the current or future state of any aspect of the business processes and underlying information systems [IIB09]. On the other hand, there is still a gap between such principles of RE and the architecture-driven research area of business information systems.

As a consequence, RE is not integrated into development process models that are built upon such domain-specific views. In the field study, this is the case for the IAF affecting the artefacts considered by the development process model *Rational Unified Process* (RUP) [KK03, JBR99]. Neither the architecture framework, nor the development process model define domain-specific methods and techniques for

1.1 Problem Statement

a company-wide RE, nor do they integrate the results into the development life cycle.

Missing Support for volatile Project Environments. The missing integration of RE into the development life cycle influences the awareness of engineers towards possibilities and necessities in RE for producing precise results. Even if a company defines and integrates an RE process, it still does not support project-specific influences that engineers have to face at project level. Exemplary influences are the availability and technical abilities of stakeholders, or constraints onto used specification structures.

In our field study, we discovered 31 project parameters affecting the RE process. Many of those parameters arise from the closeness of RE to customers capabilities and process models used, making the RE process itself highly variable. A far more complicating aspect is that many influences arise during the execution of RE, as many things are not clear from the beginning of a project [BPKR09]. The project parameters pervade therefore not only the set-up, but also continuously shape the performance of RE activities and techniques used.

Hence, the possible and necessary preciseness and quality of requirements specifications is determined by a variety of project-specific parameters. They complicate a systematic RE process within volatile project environments.

Resulting Solution Orientation in RE. We discovered different strategies in which RE processes are executed in response to individual project parameters. We subsequently explain two strategies: *Solution Orientation* and *Problem Orientation*. Both strategies are reflected in the degree of completeness of the created artefacts as a result of a missing (domain-specific) process integration and volatile project environments.

Although there is no common understanding and agreement on the term *solution orientation* yet, it is understood as the approach in which requirements remain on a high level of abstraction (def. in Sect. 2.2.3.2) and are directly distorted by solution ideas. Further development activities then act according to incomplete requirements artefacts. This increases the risk of delivering IT systems (def. in Sect. 2.3), which qualitatively do not meet the expectations of customers and, thus, do not support their organisations as expected. Instead, we consider *problem orientation* as the approach of a continuous approval process of problem statement, refinement, and problem solving, in which requirements are analysed and refined according to the customer's needs. In Sect. 4.2.3, we will define the term "solution orientation" and the related term "underspecified artefact" according to the domain-specific interpretation of our study results.

The results of the field study show that only 50% of the analysed projects performed a problem-oriented execution strategy. This manifests the idea that there still is a tendency to solution orientation, as it was the case back in the late 1970's, where 46% to 50% of analysed IT development projects didn't pay sufficient attention to RE [Boe06, Hos87, Boe79]. Solution orientation, however, can be performed on purpose, e.g., in order to increase the process efficiency.

Still, although we discover no significant losses in the process efficiency within the different execution strategies, none of the study participants showed awareness of having made an explicit decision on whether to act in a solution-oriented way or not. This unawareness can be traced back to the lack of understanding of engineers towards the necessities and possibilities in RE and project-specific influences that result in an uncontrollable development process.

Summary of Problem Statement. A major outcome of the field study is that, at organisational level, RE is not integrated into the domain-specific development process models. In addition to missing company-specific RE reference models that would support knowledge transfer among different projects, project-specific influences make the process itself highly variable. Volatile project environments barrier the appropriate choice of methods and their application and, thus, harden a systematic decision about which artefacts to produce in RE and about how to produce them in a consistent manner. An observable consequence of both the missing company-wide reference model and the diversity of the project influences is that engineers often act in a solution-oriented way what leads, in turn, to incomplete artefacts.

Although this problem is now understood, yet missing is a systematic and integrated RE approach that effectively copes with uncertain project situations.

1.2 Research Objectives

The first step to solve the problems stated above lies in the definition and integration of RE techniques, which capture the basic concepts of an application domain. A domain-specific RE reference approach supports, at project level, an awareness of producing domain-specific RE artefacts in a complete and consistent manner, since the approach makes explicit implicitly necessary knowledge on domain-specific concepts.

The next step is to provide guidance in the flexible customisation of the reference approach at project level. A flexible customisation considers the systematic creation of RE artefacts in an appropriate degree of completeness in direct response to individual project parameters, including ones that arise during project execution. Appropriateness in the degree of completeness means, habitually speaking, to act either in a problem-oriented or a solution-oriented way, when possible and when necessary, while having awareness of the consequences of potentially underspecified artefacts to other development activities. We refer to this appropriateness as *balanced problem orientation*. Obviously, the approach must precisely define when requirements are underspecified and the process therefore is solution-oriented.

Based on those ideas, we now formulate our research objectives. According to the necessity of integrating isolated RE techniques into company-specific development process models and according to the various influences that engineers have to face in volatile project environments, we aim at developing an RE approach that

1. includes an RE reference model as a model blueprint of the basic concepts of an application domain (def. in Sect. 2.5.1.1) supporting awareness of syntactically consistent and complete artefacts.
2. gives guidance to
 - a) integrate, at organisational level, the isolated reference model into the development life cycle by establishing associations between the elements of the reference model and the elements of a development process model.
 - b) customise, at project level, the reference model according to individual project characteristics. This means to assist the creation of RE artefacts in direct response to project parameters, which affect the completeness of the artefacts and to additionally support the reflection on the consequences of potentially underspecified artefacts (a solution-oriented process) on further development activities.

In the following, we briefly summarise and discuss available literature with respect to our research objectives, before defining our contribution in Sect. 1.4.

1.3 Related Work

The difficulties in designing, integrating, and customising a domain-specific RE process are recognised. However, the problem is not yet solved. One factor to be taken into account is that the principles of customisation depend on the underlying philosophy of available approaches. We distinguish between an activity-based and an artefact-based philosophy. In the following, we discuss both philosophies and selected approaches with respect to our research objectives³, before introducing the contribution of the thesis. In Chp. 3, we further analyse the philosophy of artefact orientation in detail with respect to our research objectives. In Sect. 2.6, we further discuss fundamentals in the field of customisation and the setting of our contribution in context of related work.

1.3.1 Activity Orientation

Activity orientation is based on the principle of defining a concrete process by a set of methods to be performed in a particular order by a specific set of roles. Each method provides a construction procedure to combining description techniques for a particular purpose [NE00]. Exemplary RE-specific techniques are, for example, given by Doerr et al. [DKVKP03, DKK⁺05], or in the area of agile methods, e.g., described in [PEM03, SS06]. The latter emphasise the specification of requirements according to a cost-benefit notion [KR97] implying an efficient and customer-oriented process in terms of value-based software engineering principles, as proposed by Boehm [Boe03]. Each of those techniques is then placed into a particular sequence of application and used to specify the RE results in previously defined specification documents (artefacts) [BWHW05].

Prominent examples in this area are development process models, which are based on the *Software & Systems Process Engineering Meta-Model* (SPEM) [OMG08], such as the *Rational Unified Process* (RUP) [KK03]. However, while these approaches offer the necessary elements to define a process, the customisation of the process for a particular project is barely described and left to the expertise of project participants.

As a response to this shortcoming, *Situational Method Engineering* [Bri96, THV97] provides approaches to select and combine methods from a repository. This area can further be complemented by (*content-centric*) *Decision Support Systems* [RPA⁺01, NTR05], which contribute approaches to select, classify, and rate a set of alternatives in the choice of methods (and description techniques) according to project parameters. Still, although the importance of a well-defined artefact model is recognised in this activity-based area [FBB09], the definition of artefacts, their contents, and their dependencies is not in scope of available approaches. Braun et al. [BWHW05] discovered that only 50% of the analysed approaches include an artefact description at all, while the approaches that include an artefact description reduce the artefacts to an (optional) outcome of self-contained and interconnected methods that produce the artefacts.

Hence, approaches in the activity-based area do neither define the contents of the artefacts, nor the relationships between the artefacts. The relationships between the artefacts are only (implicitly) defined via the relationships between the activities to which the artefacts are coupled. Activity orientation thereby does not give guidance in the creation of syntactically complete and consistent results, since available approaches emphasise the selection and combination of methods rather

³ The content of the following related work summary is based, in part, on our contribution in [MFLPW11].

than their integration and application considering their syntactic compatibility and the consistency in the resulting artefacts [THV97].

Available contributions that guide the (pre-)selection of a specific sequence of methods (e.g., [CCR⁺95, IIB09, JE03, AW05a, RSM95]) thus have shortcomings in tackling the problems stated above due to their emphasis on activities and methods. At organisational level, they do not support a systematic process integration. At project level, they not support the reflection on project characteristics with respect to the creation of artefacts in a particular syntactic quality. The project-specific combination of methods is additionally hampered by the diversity in the project parameters and by the variability in the RE processes, especially because many decisions cannot be taken before executing the process.

In summary, activity orientation does not satisfy the demand for an integrated RE approach that guides in the creation of syntactically consistent artefacts while supporting the necessary flexibility during this creation. This is also reflected in the absence of empirical work in this area. Despite available case studies that analyse the application of isolated methods, further studies, which consider comprehensive development process models, put emphasis on the resulting process rather than on its customisation and application [PPLB07].

1.3.2 Artefact Orientation

A possibility to achieve our objectives is to concentrate on the artefacts rather than on the way of creating them. This leads to process-neutral RE approaches, which follow an artefact-based philosophy.

The idea of artefact orientation consists of defining a reference model of all artefacts and their dependencies. An artefact is seen as a deliverable used as input, output, or as an intermediate result of a process and abstracts from the methods used to modify the deliverable. Since artefact models are defined independent of the variability in the process, they facilitate a process integration and a customisation to the needs of individual project environments. At organisational level, they ease a process integration, because they enable us to establish associations between the artefacts of RE and the artefacts of further development activities. At project level, they support a flexible process, since the actual process is defined by agreeing on a set of artefacts to be created by a particular role and to be delivered when reaching a particular milestone. The diversity in the processes is thus reduced to the dependencies between the artefacts themselves without having to take into account the complexity of differing processes. Hence, artefact-based approaches are designed to guide in the creation of precise artefacts while offering the necessary flexibility during their creation.

However, while the area of activity orientation provides via available meta models a common agreement on the principles of corresponding approaches, yet missing is a common agreement on the structure, the syntax, and the semantics of artefact-based approaches. In literature, we find different interpretations of the concept *artefact*, depending on the area of application. Available artefact models can, however, be grouped into following categories:

1. *Structure Models* that include a basic description of general topics to be considered within a structure reference of documents or data sets (e.g., as given by the *IEEE Std. 830-1998 Recommended Practice for Software Requirements Specifications* [IEE98]).
2. *Generic Content Models* that define a reference of the content with semi-formally structured checklists or guidelines to be considered without taking into account a particular application domain (e.g., as defined by the *Volere*

1.4 Contribution

Requirements Specification Templates [RR07] or by the *Requirements Engineering Reference Model* (REM) [GBB⁺06, BPKR09]).

3. *Domain-specific Content Models*, mostly represented as detailed data models as done in [SFGP05] and known as domain-specific concept models. These models characterise the concepts of a particular application domain, reflected in the elements and their relations used in corresponding description techniques (see also Schätz [Sch08a, SPHP02]).
4. *Integrated Modelling Theories* that define, mostly via mathematical models, a semantic foundation of given description techniques (e.g., *Focus* [Sch01]). Since the elements to be produced within a project are all defined according to the mathematical model and its particular syntax, syntactic consistency is enforced.

In fact, the definition of an artefact model and the understanding of the term *artefact* depends on the intended purpose of the model [ER03]. Following the categorisation from top to bottom, each of the given examples implies a higher degree of precision in the design of an artefact model with respect to the structure and the domain-specific content of the specification documents from which the models abstract. The higher the degree of precision, the higher the support of syntactic completeness and consistency, at the same time, the lower the degree of flexibility, since the artefact models do not allow for variations in the creation of the artefacts' contents.

Available development process models, which offer a customisation approach on the basis of an artefact model, consider artefact models to be structure models, because this abstract view on artefacts allows for the necessary degree of flexibility during customisation. A prominent example is given by the *V-Modell XT* [FHKS08, VMX], a German standard for IT development projects. However, such approaches neglect the artefacts' domain-specific contents and, thus, they do not support the continuous content creation of the artefacts in response to project parameters, which impact on this creation. Available approaches, which take into account the artefacts' contents, mostly consider artefact models to be domain-specific concept models. Examples can be found in the area of model-based development (see, e.g., [Sch08a, SFGP05, ZYCJ06, BW07, KKZB08]). Such approaches support the creation of syntactically consistent and complete artefacts. However, at the same time those models become complex and, thus, they hamper the process integration and the customisation at project level.

Therefore, although artefact orientation is a promising solution to tackle the shortcomings in activity-based approaches, yet missing is an artefact-based customisation approach for RE that supports an integrated and domain-specific RE process guiding the flexible creation of precise artefacts.

1.4 Contribution

We contribute an *artefact-based customisation approach for requirements engineering*, which yet is missing in literature. In response to our research objectives, we thus make use of the artefact-based philosophy to establish an integrated and customisable RE approach that guides in the flexible creation of precise results.

Figure 1.2 illustrates the single contributions of the thesis in grey coloured shapes and, on the right side, the main addressees of these contributions. The contributions are organised according to three levels of abstraction (M2 to M0), which are further discussed in Sect. 2.5.2. The contributions are:

1. A *Meta Model for Artefact Orientation* that defines, according to our research

objectives, the abstract syntax of our RE approach relying on the artefact-based philosophy.

2. An *Artefact-based Reference Model for Business Information Systems Analysis (BISA)* that defines one interpretation of the meta model for RE in the application domain of business information systems.
3. An *Artefact-based Customisation Approach* that provides means to perform a process integration of the artefact-based reference model into a development process model and to customise the integrated reference model at project level.
4. Two *Case Studies* (not depicted in the figure), which evaluate our contributions in a qualitative manner.

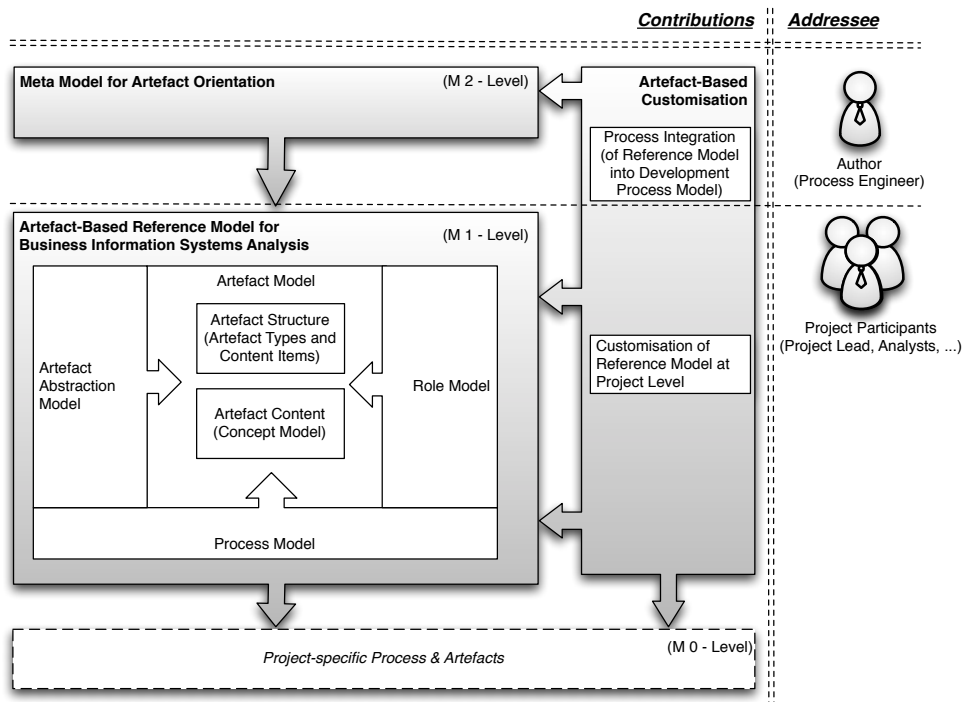


Figure 1.2: Overview of the contributions of the thesis.

In the following, we introduce our contributions and their setting in context of related work.

Meta Model for Artefact Orientation

Due to the missing common agreement on artefact orientation, we infer from our research objectives a *meta model for artefact orientation*. This meta model defines the constructs and rules of a domain-specific, artefact-based development process model with a particular focus on RE. The abstract syntax constitutes structure and semantics of artefact orientation (e.g., the artefact's concept and the relation to the role's concept). The model is constructed according to our research objectives and aims, for each domain-specific interpretation, at supporting process integration and, at project level, customisation of the domain-specific RE approach. The addressee of this contribution is therefore the author, respectively process engineer, being responsible for the development and maintenance of a development process model considered conformant to the meta model.

1.4 Contribution

Setting in Context of Related Work. We already discussed in Sect. 1.3.2 that related work in artefact orientation so far focusses on single potentially differing objectives affecting the notion of artefact orientation. Our meta model, developed as part of a foundation analysis, thereby defines according to our research objectives a notion of artefact orientation. It overcomes the missing agreement on artefact orientation and unifies different principles and views on artefact orientation given in literature with respect to our particular research objectives (see also the research method in Sect. 1.5).

Artefact-based Reference Model for Business Information Systems Analysis

We contribute an interpretation (instantiation) of the meta model for RE in the application domain of business information systems. The addressees of the reference model are project participants. The *artefact-based reference model for business information systems analysis* (short: BISA) covers the logically structured sub-models defined in the meta model in response to our research objectives: an artefact model, a (team) role model, a process model, and an artefact abstraction model.

In particular, the artefact model is defined by combining two self-contained artefact views discussed in Sect. 1.3.2. First, we define a *structure model* of the RE artefacts via a taxonomy, what is motivated by the experiences gained from the artefact model in the V-Modell XT. This structure model includes an abstract view on documents or data sets and supports flexibility during process integration and customisation, which is not in scope of pure concept models⁴. We embed into this structure model a *content model* that we represent as a concept model (data model). This concept model is motivated by the experiences gained in the area of model-based development approaches. In contrast to available concept models, however, it is specifically elaborated for RE and captures the basic concepts of the application domain by integrating isolated description techniques for the analysis of business information systems. Although we do not define the content via a mathematical model, e.g., to effectively cover a broad spectrum of modelling concepts (including also non-functional ones), we rely in our concept model, where possible, on available modelling theories to ensure the validity of the concept model for the application domain (see also the research method in Sect. 1.5). Thus, the degree of detail varies in the defined concepts depending on the approaches from which the concept model abstracts (see also Chp. 3 for further information).

We complement the artefact model by including possible *notations* for the representation of the concepts and by defining the interfaces for a potential process integration, e.g., by defining which artefacts serve as an input for specific project management activities. The *artefact abstraction model* characterises domain-specific levels of abstraction over which artefacts are produced, modified, and refined and / or decomposed. This is, for example, necessary to support the detection of under-specified artefacts (incomplete and / or remaining at a certain level of abstraction) and, thus, to infer a domain-specific interpretation of the term *solution orientation*.

Hence, the artefact-based reference model offers a self-contained basis, which can be integrated into development process models and which can be customised at project level. In contrast to existing RE approaches, it thus supports flexibility for customisation while offering guidance on the basic concepts of an application domain and, thus, on the creation of syntactically precise results.

⁴ Guiding questions are for example: “What elements of the concept model have to be considered within the task *Specify Use Case*? What elements of the concept model belong to the use case descriptions within the requirements specification?”

Setting in Context of Related Work. The artefact-based reference model unifies two self-contained artefact views – so far being treated in an isolated manner in available approaches – and puts those views into context of a customisable development process model by defining, e.g., roles and milestones.

On the one hand, we structure the artefact-based reference model motivated by the experiences gained from the area of artefact-based development process models reflected in approaches like the V-Modell XT. On the other hand, the artefact model includes a concept model specifically designed for RE in the application domain of business information systems, which by now is missing in literature. To this end, we develop a concept model by abstracting from domain-specific description techniques and theories, and by integrating those approaches on the basis of a unified artefact model (see also the introduction of the research method in Sect. 1.5 and Sect. 3.3.3 introducing the validity procedure during the development of the artefact model).

Artefact-based Customisation Approach

The artefact-based customisation approach defines mechanisms to perform a process integration of the artefact-based RE reference model into development process models and to customise the reference model at project level. At project-level, we provide, in particular, assistance on the (static) set-up of a project by generating exemplars of the sub-models of the reference model and on decision taking with respect to the continuous content creation of the artefacts in response to project parameters that affect the artefacts' completeness. During the content creation, we assist the reflection on potentially underspecified artefacts and the consequences w.r.t. further activities that rely on those artefacts. Hence, we support awareness of a domain-specific process-integrated RE process and, thus, we enable a balanced problem orientation.

Setting in Context of Related Work. Since the approach for a process integration of a reference model into a development process model depends on structure and content of the reference model to be integrated, our process integration approach does not directly relate to available approaches.

To define the customisation approach for applying our integrated reference model at project level, we investigate related work, e.g., in the area of situational method engineering, and transfer, in parts, available approaches and principles to the artefact-based philosophy. Since our artefact-based reference model includes a notion of RE contents, we can assist – in contrast to available customisation approaches – an appropriate decision taking in RE during project execution and the continuous content creation of the artefacts in awareness of potentially underspecified artefacts. Thus, we tackle the shortcomings in available customisation approaches. Further information on the relation of our contribution to existing customisation approaches is given in Sect. 2.6.

Case Studies

We finally contribute an evaluation of our contributions with respect to our research objectives via two case studies. The first case study evaluates the process integration of our reference model into the V-Modell XT. The second case study evaluates the application of our reference model in an industrially hosted project environment.

1.5 Research Method

Setting in Context of Related Work. To the best of our knowledge, there is no existing comprehensible case study considering, at organisational level, a process integration, or, at project level, the application and customisation of an artefact-based or an activity-based approach. Hence, our case studies close this gap in literature. Further information is given in Chp. 6.

1.5 Research Method

The contributions of this thesis tackle the shortcomings of existing approaches stated above by analysing, extending, and finally integrating a large extent of available work on the basis of an artefact-based philosophy. The artefact model, for example, abstracts from isolated description techniques and modelling theories and integrates them on the basis of a unified model. Hence, an interesting and at the same time important question is how to ensure a scientifically valid research method and, thus, the validity of the contributions themselves. For this reason, we now describe the chosen research method, which is depicted in Fig. 1.3.

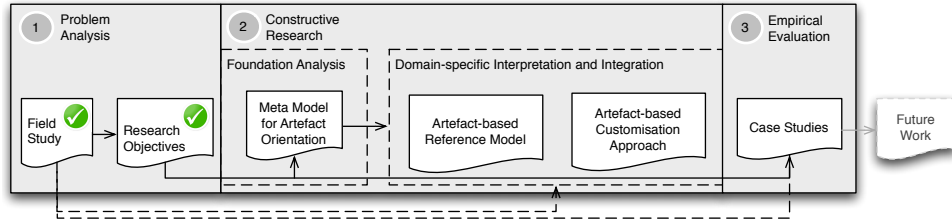


Figure 1.3: Research method chosen in the thesis.

Problem Analysis. The contributions of the thesis rely on a problem analysis previously performed as part of a field study [MFWLB10, MFWL⁺12] (see also the problem statement in Sect. 1.1). In this exploratory study⁵, we seek for observations regarding the definition and execution of RE processes in an industrial context in order to formulate appropriate research objectives and to account for their relevance. Hence, the field study is not part of our contribution, but defines a necessary empirical basis on which our contributions rely. The reason for conducting the field study is that available studies put either emphasis on the maturity of RE processes [BHR05, BHR03, SR05, BWSV08], relating factors that influence the success of process improvements [Gro, BEJ06], or they analyse used techniques, and project characteristics [BA98, CNV09, EEM95] in relation to the projects' success in general [Gro, BEJ06]. Thus, they give a coarse-grained view on general problems in RE processes. However, they give no understanding of (1) aspects important to a particular application domain, of (2) project parameters affecting the RE process in a particular way, and of (3) particular execution strategies (and their appropriateness) used in response to the project parameters.

Therefore, we investigated those factors as part of the problem statement to infer our research objectives, but also to reuse selected study results (illustrated in Fig. 1.3 with dashed lines) in terms of

1. determining artefacts that are of interest for the chosen application domain (see also Sect. 3.3.3 describing the development of the artefact-based reference model),

⁵ We use the term according to Scapens [Sca04] to indicate its intended purpose of a "preliminary investigation".

2. categorising and evaluating different RE execution strategies and their effects on the completeness of the created artefacts (see also Sect. 4.2.3 introducing different strategies for creating the artefacts), and
3. investigating project-specific reasons that lead to particular execution strategies by extracting project parameters (see also Sect. 5.3 introducing the artefact-based customisation approach considering the customisation towards a particular strategy according to project parameters).

Constructive Research. Based on our problem analysis, we choose for a constructive research approach, i.e., a research procedure intended to solve real world problems (discovered in the field study) by making a contribution to the theory in the corresponding discipline [Luk00]. We design this procedure in two steps.

As a first step, we conduct a theoretical *foundation analysis*. We analyse artefact orientation with respect to activity orientation and discuss possible notions in artefact orientation in dependency to differing objectives followed when establishing an artefact-based approach. We then infer according to our particular research objectives a meta model for the philosophy. This meta model overcomes the missing agreement on artefact orientation and unifies different principles and views on artefact orientation given in literature with respect to our particular research objectives.

As a second step, we perform a *domain-specific interpretation* of the developed meta model for a particular domain of application. During this interpretation, we *integrate* available and established, but isolated approaches and principles on the basis of the artefact-based philosophy reflected in our meta model.

The domain-specific interpretation of the meta model for the domain of business information systems is performed, in major parts, in an industrially hosted research environment in which we developed and continuously evaluated the previous version of the BISA approach *Quasar Requirements* [MFC09] (see also Sect. 3.3.3). Thus, although this model is not the only possible domain-specific interpretation of the meta model, we ensure its validity for the application domain. As an additional validity procedure holds the definition of the artefact model itself, since it abstracts from established description techniques and, where possible, from modelling theories, both considered during our previously performed field study as relevant for the application domain. This supports the completeness of the concepts for the domain. Regarding the customisation approach, we consider, for example, the introduced areas of situational method engineering and decision support systems. We transfer the basic thoughts of those activity-based approaches to the artefact-based philosophy (according to our meta model) in order to tackle their shortcomings with respect to our research objectives.

Empirical Evaluation. We gained practical experiences during the development and evaluation of our artefact-based reference model in an industrial environment. However, we still have no evidence on whether we achieved our research objectives and whether our artefact-based customisation approach tackles the shortcomings in given (activity-based) approaches. A further influencing factor is given by the absence of empirical studies in the area of activity orientation and artefact orientation, whereby the choice for artefact orientation is based, in parts, on arguments and practical experiences gained in related areas.

Therefore, we perform two case studies, which evaluate our contributions in a qualitative manner following our proposed customisation procedure. The first case study evaluates the approach to perform a process integration. The second case study evaluates our project-specific customisation approach in an industrially

1.7 Previously Published Material

hosted project environment considering a comparison to the activity-based RE approach previously used in the same project setting. Hence, we do not only close a gap in literature with respect to case studies in the area of artefact orientation and activity orientation, but we also underpin our argumentation with an empirical basis. In addition, we use the results of the case studies to direct future work.

1.6 Outline of the Thesis

The remainder of this thesis is organised as follows. Chapter 2 describes the fundamentals and related work in the areas of our contribution. In Chp. 3, we contribute the foundation analysis leading to the meta model for artefact orientation. We conclude the chapter with a description of the validity procedure performed during the instantiation of the meta model, i.e., during the development of the artefact-based reference model for the application domain of business information systems. This artefact-based reference model is presented in Chp. 4. In Chp. 5, we then define the artefact-based customisation approach. The contribution is then evaluated in Chp. 6 with two case studies, before concluding the thesis in Chp. 7 with a summary of our results and an outline of future work. Finally, we define in appendix A a glossary of relevant terms used in this thesis.

1.7 Previously Published Material

The material covered in this thesis is based, in part, on our previous contributions in [MFWLB10, MFWL⁺12, MFK09, MFC09, MFPKB10, WMFIL09, LWMFB10, BLMF⁺10, IHMFJ09, BFI⁺09, MFLPW11].

In particular, the previously performed field study can be found in [MFWLB10, MFWL⁺12]. An analysis of fundamentals in artefact-based RE and the inference of an initial meta model for artefact orientation is published in [MFPKB10] and relates to Chp. 3. The artefact-based approach for the application domain of business information systems (Chp. 4), initial versions of the customisation approach (Chp. 5), and a case study on a preliminary process integration, rely on our technical report [MFK09]. This report extends our initial artefact-based reference model [MFC09], being developed, evaluated, trained, and finally used as the standard RE reference model at Capgemini TS (see also Sect. 3.3.3). A case study on the project-specific application of our artefact-based customisation approach in an industrial setting is published in [MFLPW11].

Further contributions [WMFIL09, LWMFB10, BLMF⁺10, IHMFJ09] cover selected parts of the artefact model, including quality requirements and their conceptualisation according to quality definition models, concepts of service-oriented architectures, and concepts used for risk management activities.

Additional information is provided in corresponding sections in which we refer to our previous contributions.

CHAPTER 2

Fundamentals and Related Work

This thesis contributes an artefact-based customisation approach for RE in the application domain of business information systems. We thus relate with our contribution to several areas, including

1. artefact orientation and its interplay with development process models (related to our meta model for artefact orientation in Chp. 3),
2. requirements engineering and its interpretation for the application domain of business information systems (related to our artefact-based reference model for business information systems analysis in Chp. 4), and
3. customisation of development process models (related to our artefact-based customisation approach in Chp. 5).

In each of those areas, we consider a set of fundamentals and related work for our contributions. In Sect. 2.1, we give a detailed overview of the related work areas, their interrelations with our contributions, and an outline of the chapter at hand. In subsequent sections, we discuss the different areas in detail. The content of this chapter is based, in part, on our contribution in [MFK09].

At the end of this chapter, the reader will have an overview of the areas related to our contributions: RE in general, the application domain of business information systems, and an integrated view of both. A discussion of development process models and their customisation rounds out the fundamentals and related work needed in the context of this thesis.

Contents

2.1	Outline of Fundamentals and Related Work	16
2.2	Requirements Engineering and Management	17
2.3	Fundamentals in Business Information Systems	26
2.4	Requirements Engineering for Business Information Systems .	32
2.5	Development Process Models	34
2.6	Customisation of Development Process Models	45
2.7	Introduction of a Running Example	57

2.1 Outline of Fundamentals and Related Work

As stated in the introductory part of this chapter, we discuss different areas that are related to our contributions. Figure 2.1 illustrates these areas and to which contributions they relate (depicted with the dashed box in the middle of the figure). We distinguish between the area of *requirements engineering and its interpretation for business information systems* and the area of *development process models and their customisation*. The first area relates to our contribution in Chp. 4, which defines an artefact-based reference model for the application domain of business information systems. The second area relates to our contributions in Chp. 3 and Chp. 5. It relates to the meta model for artefact orientation (on which the reference model relies) and to the artefact-based customisation approach.

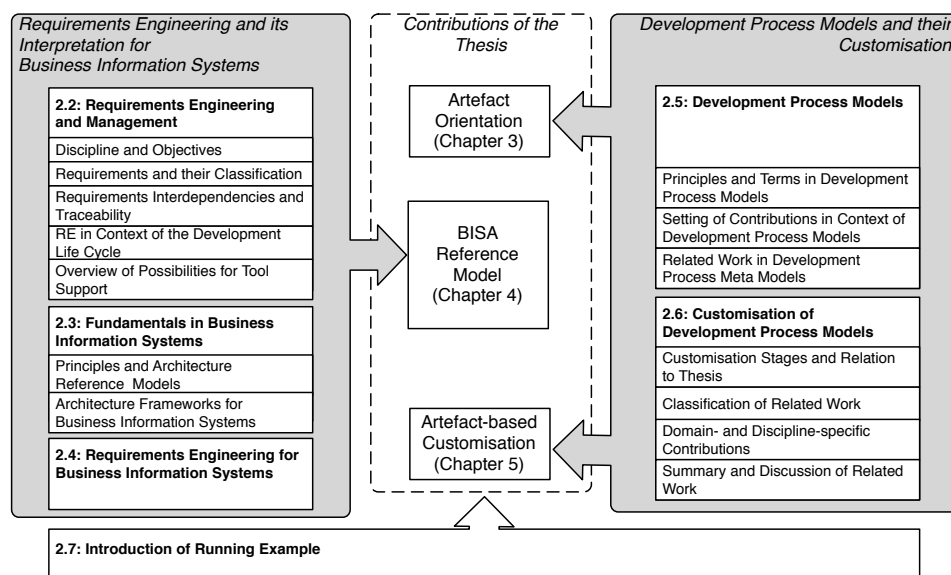


Figure 2.1: Outline of related work and relation to contributions.

In the following, we describe the outline of the chapter, structured according to the introduced areas.

Requirements Engineering and its Interpretation for Business Information Systems. In Sect. 2.2, we lay the foundation in requirements engineering and discuss selected topics in direct relation to our contribution in Chp. 4. After giving an overview of the discipline and its objectives, we discuss the topic of requirements classification, interdependencies between requirements classes, traceability, including also refinement over levels of abstraction, and the interdependencies between RE and the development life cycle (used for a process integration). These topics directly relate to the artefact model, which is introduced in Sect. 4.3. We conclude this section with an introduction of tooling in requirements engineering and management to lay the foundation for the tools used in the case studies of Chp. 6. In Sect. 2.3, we summarise the fundamentals in business information systems, e.g., by introducing different notions of an architecture and available frameworks for the construction of architectures. These fundamentals are not discussed in direct relation to our contributions. The introduction of the fundamentals, however, serves in the next Sect. 2.4 to sketch a first domain-specific interpretation of RE,

i.e., to sketch to what extent we see RE to be covered in frameworks for describing business information systems. We then take this interpretation in Sect. 4.1 as a basis to define the scope of the artefact-based reference model for the application domain.

Development Process Models and their Customisation. In Sect. 2.5, we analyse the fundamentals and discuss related work in the area of development process models. To this end, we first introduce the terminology used in the thesis. After discussing principles of modelling and meta modelling, as they are used in the context of development process models, we define the setting of the single contributions of the thesis. We conclude this section with an overview of related work in development process models and a discussion with respect to our contributions. We take a particular focus on the meta model of the V-Modell XT, because we refer to the V-Modell XT as a concrete example for analysing the area of artefact orientation in Chp. 3, and, in Chp. 6, as a concrete case study for integrating our artefact-based reference model into a development process model.

In Sect. 2.6, we then discuss the last area related to our contribution: the customisation of development process models. We introduce relevant terms and principles, and analyse related work with respect to our contribution of an artefact-based customisation approach in Chp. 5.

In Sect. 2.7, we finally introduce a running example to which we refer in our contributions.

2.2 Requirements Engineering and Management

In this section, we discuss the principles, terms, and related work in requirements engineering, and show the relations to our contribution. We first characterise the discipline and its objectives. We then discuss the main activities in RE and how requirements can be classified. Afterwards, we show what relations are typically used for the description of the interdependencies between the different requirements classes. We conclude this section with a discussion of the boundaries of the discipline within the overall development life cycle, and discuss possibilities for tool support.

2.2.1 Discipline and Objectives

Software engineering, as a systematic and cost-effective approach to software development projects [IEE90], can be divided into different phases or disciplines¹, such as requirements engineering, design, implementation and testing.

Requirements engineering (RE) constitutes activities within the initial and thereby most critical phase of development projects, because requirements are the critical determinants of software quality [AW05b]. RE aims, as a branch of software engineering, at

1. capturing and specifying the problem space, i.e., “real-world goals for functions of and constraints on software systems” [Zav97], in a proper level of detail. Based on the specified requirements, further activities of the development life cycle act according to the stakeholders’ needs.

¹ The used terms depend on the chosen development process model, see also Sect. 2.5

2.2 Requirements Engineering and Management

2. continuously administrating the specified requirements (and changes) over their whole life cycle.

The objectives of the discipline are therefore two-fold. RE concentrates, on the one hand, on engineering the problem space via a set of activities and, on the other hand, on management activities. Both requirements engineering and management are introduced in the following.

2.2.1.1 Requirements Engineering

According to Geisberger et al. [GBB⁺06], requirements engineering addresses all aspects of capturing, analysing, deciding, structuring, and validating requirements on products. RE is seen as an interdisciplinary set of manifold activities involving stakeholders from different domains [Gei05]. In literature, there exist several frameworks and approaches to combine activities in a notion of a development process model (def. in Sect. 2.5). Aurum et al. give an overview of such frameworks in [AW05b], including linear, incremental, non-linear, and spiral models.

It is, however, recognised that the effectiveness and efficiency of a process in RE relies on an iterative approach for problem statement and problem solving. It relies on a continuous elaboration of a solution and the description of the problems in several iterations in which the problem space is narrowed down with continuously made design decisions (see also Conklin et al. [CW97]). As a consequence, Sommerville et al. [KS98, SS97] and Loucopoulos et al. [LK95] describe the RE process to be iterative and cyclic by nature [AW05b].

We therefore use according to these views the following definition for requirements engineering:

Definition: Requirements Engineering

Requirements Engineering (RE) aims at describing a problem space as comprehensively as possible by comprising iterative and systematic approaches to define a requirements specification aligned to the needs of all relevant stakeholders.

Obviously, the definition is generic so that a domain-specific interpretation is necessary. Such an interpretation is given in Chp. 4.1 by introducing *business information systems analysis* (BISA).

2.2.1.2 Requirements Engineering Activities

The RE process is structured into four major activities, independent of their alignment (e.g., as part of an iterative process) [SS97, Wie03]:

- *Requirements Elicitation* considering the “trawling for requirements” [RR99], including communication, prioritisation, negotiation, and collaboration with all relevant stakeholders [ZC05].
- *Requirements Analysis* considering refinement of requirements over several levels of abstraction (see Sect. 2.2.3), and classification of requirements (see Sect. 2.2.2) [GBB⁺06].
- *Requirements Documentation* considering the modelling of requirements and the structured documentation according to, for example, document standards like the *IEEE Std. 830-1998 recommended practice for software requirements specifications* [IEE98].
- *Requirements Validation and Verification* considering quality assurance of requirements specifications w.r.t. characteristics like completeness, correctness,

and unambiguousness [IEE98], and the procedure of determining whether the final system complies with the requirements specification².

Each of the activities includes different methods and description techniques, like ones used in the context of goal modelling. Similar as it is the case for the term *requirements engineering* itself, the interpretation of RE activities and description techniques used depends on best practices, respectively the organisational culture of a particular company, and on the application domain. Based on a practitioner survey, the *ReqMan Process Framework* [DKOA06] defines, e.g., for each of the activities a set of description techniques. Similar work is performed by Zowghi et al. [ZC05]. They give an overview of different elicitation approaches and classify corresponding techniques that can be used.

Relation to Thesis. In the thesis at hand, we concentrate on an artefact model for the application domain of business information systems. Section 2.3 introduces the application domain before describing the interrelations between RE principles and the ones in the application domain in Sect. 2.4. According to the inferred scope, we then interpret the activities of the discipline and introduce the relevant artefacts to be created in Chp. 4 (taking into account our previously performed field study; see Sect. 1.5). Since these artefacts abstract from available domain-specific description techniques, we introduce related work as part of the the artefact model.

2.2.1.3 Requirements Management

Requirements management (RM) is considered as the procedure to administrate the outcome of engineering activities, i.e., to administrate requirements artefacts over their life cycle [FGP04]. This administration includes activities, like:

- Execution and control of change management procedures, including traceability, impact analyses, and progress control (see also [JL05]).
- Determination, assessment, and mitigation of risks that relate to requirements [Isl09].

To perform such activities as part of RM, it demands for structuredness of requirements specification documents, consistency in the content of the specifications, and contents that include information going beyond the requirements themselves, i.e., attributes such as the priority of requirements [AW05b].

Hence, the effectiveness of requirements management activities relies on the structure of requirements specifications according to chosen requirements classes, on a defined set of interdependencies, and on chosen requirements attributes. We subsequently introduce the fundamentals in these areas.

2.2.2 Requirements and their Classification

The *IEEE Standard Glossary of Software Engineering Terminology* [IEE90] defines the term *requirement* as follows:

- (1) A condition or capability needed by a user to solve a problem or achieve an objective.
- (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
- (3) A document representation or a condition or capability as in (1) or (2)

² Guiding question for validation is “Are we building the right system?”; guiding question for verification is “Are we building the system in a right way?” (see also [IEE90]).

2.2 Requirements Engineering and Management

A condition or capability to be met by a system is, however, only one of many facets a requirement can have while the corresponding type of requirements (“requirements class”) is often referred as *system requirements* (like functional requirements) [Wie03]. Glinz, for example, introduces in [Gli07] a taxonomy with different requirements classes. In his taxonomy, he classifies requirements according to their different characteristics. The taxonomy includes also requirements classes, which go beyond system-relevant requirements, i.e., project and process requirements³ (see also Fig. 2.2).

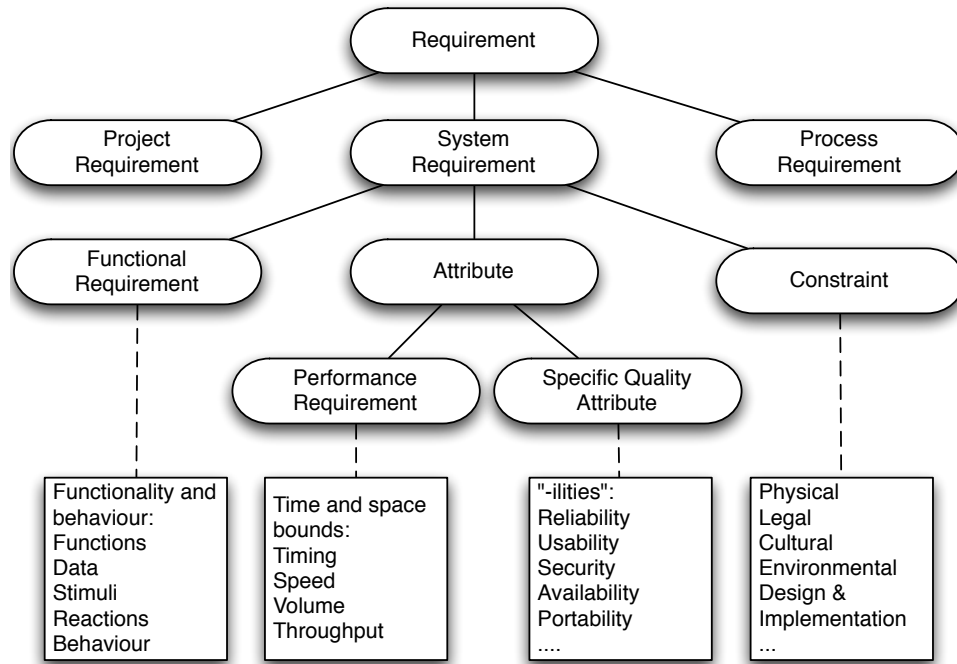


Figure 2.2: Taxonomy of requirements as proposed by Glinz [Gli07].

A similar taxonomy is introduced by Lamsweerde [vL09]. He gives a more detailed view on the notion of non-functional requirements. This view is, in turn, reduced in the depicted taxonomy to the quality attributes defined by the *ISO Std 9126 for Product Quality* [ISO03] (the “-ilities”). Each requirements class in the taxonomy additionally includes attributes used to describe context information. Context information supports during requirements’ evolution different management activities, such as requirements control and decision taking [RR99, Rup07]. Exemplary attributes are the requirements ID, their priority, their state, or the stakeholders that demand the requirement (see also [RR07]).

The classification of requirements and the choice of attributes depends on the application domain. The reason is that the classification of requirements relates to chosen description techniques, which are used for modelling the requirements’ assertions in dependency to the domain’s characteristics. This is especially true for the description and classification of quality-related requirements, whereas quality is, in general, a multi-facetted topic (see also Garvin [Gar84] and Kitchenham et al. [KP96]). Another reason for a missing common agreement on a comprehensive requirements classification is that the discipline itself is not well-bounded with, e.g., quality assurance or project management [EW05].

³ Although not directly defined by Glinz, project requirements are seen as agreements on “contractual matters” w.r.t. delivery schedules and cost [IEE98], and process requirements are seen as restriction placed on the methodology used.

Relation to Thesis. Each requirements taxonomy is, at least in parts, a specific-purpose taxonomy that considers a particular application domain. The requirements classification considered in the context of this thesis is incorporated by the artefact model of Chp. 4 and introduced according to the our application domain in Sect. 4.5.2.1. In this artefact model, we also introduce a set of requirements attributes.

2.2.3 Refinement Interdependencies and Traceability

Requirements exhibit different types of interdependencies, which influence development activities and the decisions made during these activities; for example, during change management or during testing [DP05]. Guiding questions are, e.g., if requirements make similar or contradictory assertions, or if the syntax chosen for representing one requirement is syntactically compatible with the syntax chosen for another (related) requirement.

The main purpose of capturing the interdependencies consists in the support of requirements management activities. This enables the determination of consistency in the requirements specifications, and supports activities like impact analysis.

We subsequently give an overview of the topics that are related to interdependencies, including traceability and levels of abstraction. We conclude with an overview of types of interdependencies considered in the context of the thesis.

2.2.3.1 Traceability

Traceability considers “the ability to describe and follow the life of a requirement, in both a forward and backward direction, i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through periods of on-going refinement and iteration in any of these phases” [GF94]. Traceability helps to demonstrate that each requirement has been satisfied. It furthermore ensures that each developed component of the system satisfies a requirement [Wri91]. Ramesh et al. analyse in [RJ01] the notion of traceability and synthesise and validate different reference models. They consider the term *traceability* as a “characteristic of a system and its development in which the requirements are clearly linked to their sources and to the artefacts created during the system development life cycle”. The documented links are defined as *Requirements Traces*. *Forward tracing* describes the investigation of traces forward to the artefacts that are accordingly produced. *Backward tracing* describes the investigation of traces backward to the artefacts’ rationale.

Relation to Thesis. We support traceability by documenting the different traces in a traceability matrix, presented in Sect. 4.6. As traceability always depends on the artefact model, respectively the documented dependencies between the artefacts, the traceability matrix relies on the previously defined artefact model (capturing requirements interdependencies to be traced).

2.2.3.2 Levels of Abstraction

A requirements trace captures the linkage between different requirements artefacts that result, e.g., from hierarchical decomposition and / or from refinement activities [Poh10].

2.2 Requirements Engineering and Management

In our context, *refinement* considers a relation between two concepts on different levels of abstraction. The assertion of the concept, which are refined to further concepts, can be reproduced by the collection of the latter concepts (see also Schätz [Sch08a]). We refer to refinement when analysing functional (system) behaviour. In the area of RE, however, refinement is understood in a broader sense, describing also the refinement of non-functional requirements over different levels of abstraction. We distinguish between vertical levels of abstraction and, within each of those levels of abstraction, separation of concerns by means of (modelling) views [Poh10].

Vertical levels of abstraction refer to a hierarchy of different levels of abstraction over which requirements artefacts can be refined and decomposed and is in scope of most RE approaches considering “requirements refinement” as a topic (see, e.g., [Wie03, GW06, IIB09]). Each level of abstraction includes a different perspective on the content of the requirements. Going levels of abstraction from top to bottom means to break down artefacts and enrich their content with additional details [Coc00]. Gorschek et al. [GW06] introduce with the *Requirements Abstraction Model* (RAM) four different levels of abstraction, i.e., perspectives that can be taken onto a product. The product level, as the most abstract one, including goals; the feature level including an abstract description of system functionality necessary to achieve the goals; the function level describing single functions from a user’s perspective; the component level describing the internal realisation of the requirements within a (logical) component architecture.

Wiegiers [Wie03] introduces three levels of abstraction to which he allocates specific requirements classes (see Tab. 2.1).

Table 2.1: Vertical levels of abstraction defined by Wiegiers [Wie03].

Levels of Abstraction	View	Exemplary Representation
Business Requirements	High-level business view including organisation structures and incised market	Business processes & Goals
User Requirements	View taken by end users onto the system under consideration (as a black box)	Use Cases
System Requirements	System-centric view. May include structure of systems and their internal realisation (glass box)	Functional Requirements

Business requirements are artefacts that represent organisational and strategic issues, like business processes to be supported by a system. This level is not considered by RAM at all. User requirements are artefacts that represent the requirements from a user’s point of view, such as use cases which describe intended scenarios of interaction between users and a system. The system is seen at this level of abstraction as a black box that describes user-visible (external) behaviour rather than giving information about internal realisation details. System requirements are found at the lowest level of abstraction. System requirements describe the internal realisation of a system, e.g., the component architecture and the interaction between single components.

Within each of the vertical level of abstraction, additional separation of concerns

is possible. In the context of this thesis, we realise this separation of concerns with *modelling views*. Modelling views describe aspects like structure or functionality that have to be vertically brought together in a consistent manner (also referred to as horizontal levels of abstraction, see also Schätz et al. [SPHP02]).

Relation to Thesis. In Sect. 2.3, we introduce the levels of abstraction that are considered by architecture frameworks for business information system. In Sect. 4.2, we then interpret those levels to an artefact abstraction model and allocate a set of artefacts.

2.2.3.3 Types of Requirements Interdependencies

As described in the introductory sections, dependencies between the requirements result from the relation between different classes of requirements, being captured for traceability reasons. The dependencies are given in a syntactic and / or a semantic manner. Dahlstedt et al. [DP05] analyse and classify different types of dependencies. They group the dependencies into structural dependencies, constraint dependencies, and others that relate to specific techniques, such as cost/value dependencies.

Structural dependencies are defined according to the requirements classification, e.g., functional requirements relating to quality requirements and, thus, they are syntactic dependencies. Constraint dependencies relate more to the content of the requirements, i.e., the requirements' assertions and, thus, they correspond to semantic dependencies.

Since the dependencies, like the requirements classification itself, arise from the domain of application and the description techniques used, we now reduce the dependency types to syntactic dependencies and semantic ones (see Fig. 2.3).

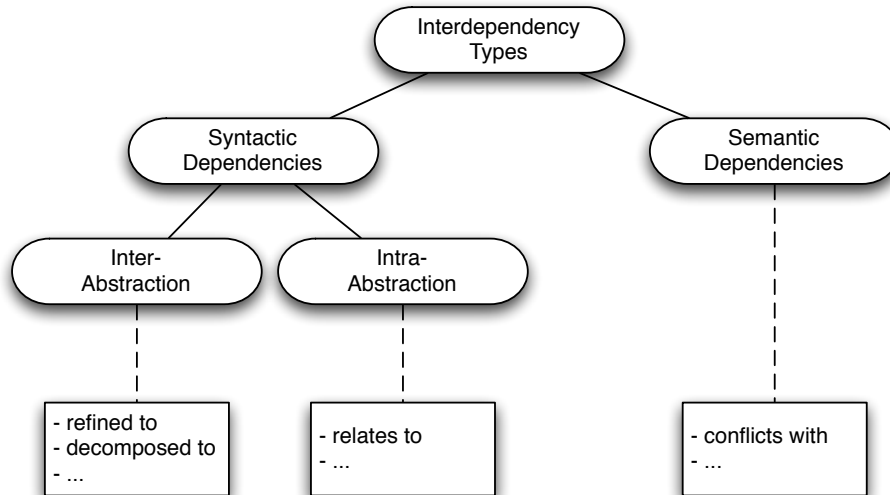


Figure 2.3: Interdependency types in the context of the thesis.

Semantic dependencies take the view of the requirements' assertion and describe, e.g., whether requirements make contradictory assertions. Syntactic dependencies result from the requirements classification and show how different classes, created with different description techniques, potentially rely on each other. Hence, each

2.2 Requirements Engineering and Management

semantic dependency demands for a syntactic dependency. According to the introduced levels of abstraction, the dependencies can additionally span different vertical levels of abstraction or they can result from the relation between two requirements classes at the same level of abstraction. We depict in Fig. 2.3 for each of the dependencies examples in the boxes.

Relation to Thesis. We focus on the definition of syntactic dependencies. The syntactic dependencies are further classified in the introduced meta model in Sect. 3.3. In Sect. 4.3.3, we give the domain-specific interpretation of the dependencies by considering the dependencies in the description techniques from which the artefacts abstract.

2.2.4 RE in Context of the Development Life Cycle

So far, we discussed requirements, their refinement over different levels of abstraction, and, in part, the transition to design activities (see Sec. 2.2.3.2). We showed that the RE activities need an interpretation for a particular application domain.

In fact, RE is to be seen as an interdisciplinary area [Gei05]. The boundaries of RE are not well defined in context of further activities of the development life cycle [EW05]. The interplay of RE with further activities of a development life cycle depends on the application domain with the underlying architecture principles and on the used development process model (see the problem statement in Sect. 1.1).

For example, it is acknowledged that requirements have dependencies on project management activities, because requirements are used, e.g., as a basis for calculations of effort. Nevertheless, how to exactly define these dependencies depends on the application domain and the associated description techniques. We can perform the calculation of effort, e.g., via function points on the basis of use cases that represent user requirements. Such an association then would reflect a domain-specific dependency, since the use cases are the necessary input for effort calculations. A further aspect to be taken into account is the followed philosophy, i.e., whether we follow an artefact-based philosophy or an activity-based one. If we focus on activities rather than on artefacts, the associations have to be established between description techniques (e.g., “define use case model” and “perform calculations of complexity”) and not between the artefacts (e.g., “use case” and “function points”).

Hence, available related work discusses RE in relation to the development life cycle either in a generic manner or as part of best practices gained from a survey. Hood et al. [HWFP07] discuss the relationship between activities and RE in a general manner; for instance, the relation to *project management*, *configuration management*, *risk management*, or *test management*. Requirements can, for example, be a source for project risks, to be taken into account in the development of corresponding risk mitigation strategies [Isl09]. Damian et al. [DC06] present a study in which they analyse the relations between RE to further activities from the perspective of process improvement in RE and its payoffs. Their findings give a more fine-grained view on the relations between RE activities and, for example, *test management*. The authors discuss, in contrast to Hood et al., the benefits of RE process improvements on, e.g., test automation.

Based on available contributions, we sum up (in a simplified manner) the dependencies of RE to general software development activities:

1. Design & Implementation
2. Quality assurance (including test management)
3. Risk management

4. Project management
5. Configuration & Change management
6. Release management & Deployment

Relation to Thesis. We develop an artefact-based RE approach and perform a process integration of the approach into a development process model. To perform such a process integration, we define the interfaces of our artefacts to activities of the development life cycle in Sect. 4.7. The process integration into a concrete development process model is shown as part of a case study in Chp. 6.2.

2.2.5 Overview of Possibilities for Tool Support

There exists a variety of tools, which support the engineering of requirements (elicitation, analysis, and modelling of requirements), and their management (administration of requirements over their life cycle). Zowghi et al. [ZC05] and Ebert et al. [EW05] give an overview of available (software) tools in these areas.

Ebert et al. [EW05] give an overview of the main features of available requirements management tools (RM tools). They highlight the importance of supporting change management and traceability for RE as an interconnected discipline. Industrial tool suppliers, such as Telelogic with Doors⁴ or IBM Rational with Requisite Pro⁵ aim with tool support at supporting both requirements management and requirements engineering. The latter is enabled by integrating RM tools with regular CASE tools for, e.g., modelling requirements by the use of the *Unified Modelling Language* (UML) [OMG10a, OMG10b].

Because RE exhibits many dependencies to further activities of a development process (see the foregoing section), we take into account a third category of tools: development process modelling tools (see Fig. 2.4).

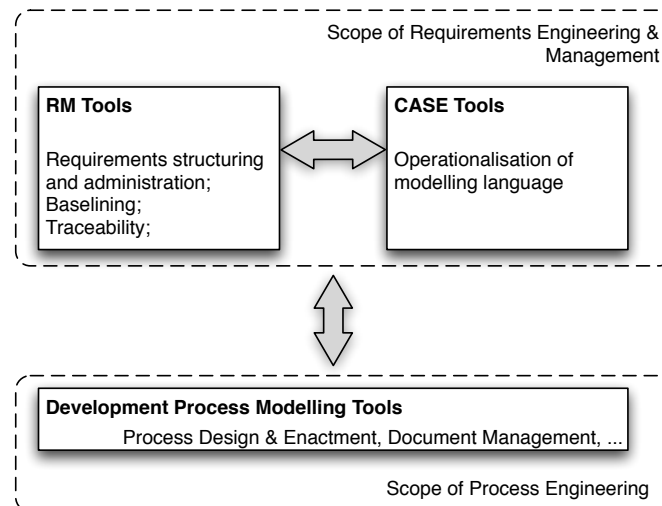


Figure 2.4: Areas of tools considered by requirements engineering as an interconnected discipline.

Tools in this area support the definition of development processes, as well as their enactment.

⁴ www.telelogic.com

⁵ www.rational.com

2.3 Fundamentals in Business Information Systems

Relation to Thesis. In this thesis, we focus on conceptualisation of a customisable RE approach. Tool support or concepts for tools are not in our primary scope. We make, however, use of available tools as part of the case studies in Chp. 6. Considering the process integration of the approach into a development process model, we make use of development process modelling tools. Considering the application of the integrated approach in a real life project, we use available CASE tools for modelling and managing requirements w.r.t. structures and contents defined in our artefact model.

2.3 Fundamentals in Business Information Systems

In this thesis, we contribute an artefact-based customisation approach for RE considering the application domain of business information systems. We thus give in this section an introduction into the application domain and discuss the necessary fundamentals.

For this, we first introduce the principles of the application domain, how they relate to the objectives of the discipline RE, and the notion of an architecture in this context. We conclude the section with the description of exemplary architecture frameworks. We use the understanding of architecture frameworks to show, in Sect. 2.4, to what extent RE is and should be covered by selected views (and artefacts) proposed by the architecture frameworks. This gives a first interpretation of RE in the context of business information systems.

2.3.1 Principles and Architecture Reference Models

An application domain considers the characterisation of a bounded area (environment) in which systems of a particular family is used. The characteristics of an application domain influence the choice of (system) modelling concepts and the semantics of their notion.

However, the characterisation and classification of a particular application domain cannot be trivially expressed, because it also depends on the view taken onto the family of systems used, usually reflected in architecture reference models. Hence, to characterise the application domain of business information systems, we subsequently discuss as a first step its basic principles

1. *IT Business Alignment*, describing how information systems shall support the business (processes) of a company, and
2. *Architecture Models*, describing how the business and the underlying systems shall be described and implemented in order to enable this support.

Both principles are discussed in the following.

2.3.1.1 IT Business Alignment

The principle *IT Business Alignment* considers, habitually speaking, the description of how systems must support a company in its “daily business”. The description of systems covers according to the *IEEE Standard Glossary of Software Engineering Terminology* [IEE90] “a collection of components organised to accomplish a specific function or set of functions”. The term *Information Technology* (IT) covers logical, technical, and technological aspects of a system (see also [BFG⁺08]). The principle thereby considers approaches to describe what capabilities (functionality and quality) systems shall offer, how they shall be structured, and what technologies shall be used in order to support a set of business processes of a company.

IT business alignment results from a methodological point of view, which aligns two (sets of) levels of abstraction: the levels that consider the business processes of a company (the “real world”) and the levels that consider the systems supporting a company in realising their business processes.

Business. The term *business* addresses the business processes of a company and non-functional aspects that are necessary to achieve some outcome of value. Business processes are described in a sense that reflects the real world as closely as possible⁶. Business processes incorporate a description of [Thu04]

- activities that are performed,
- roles that perform the activities, and
- the information that are interchanged and processed when performing the activities.

The description of business processes emphasises economically driven aspects rather than aspects of underlying systems. Further arising terms are, in this context, the one of a *value chain* and the one of a *supply chain*. A value chain describes the linkage between values, which are an outcome of performed business processes; for instance, the value to a customer that orders a travel at a travel agency, and the value to a company that offers the travels by purchasing them from another company. A supply chain, in turn, describes the linkage of business processes between companies and markets, between several companies and so on. This chain of business processes is needed to produce the goods or offer the services (e.g., to a market) that achieve the value [FSC06]. The description of supply chains and value chains supports an economically driven view onto business processes and a flexible integration of business processes, e.g., when integrating suppliers or outsourcing business processes [EHH⁺08].

Information Systems. The term *information system* considers information processing systems used to realise the business processes of a company and its supply chains; for instance, a travel ordering system, necessary to support the business processes of a travel agency. We use the term *Information Systems (IS)*, *Business Information Systems*, and *IT Systems* as synonyms and refer to information processing systems, which are used to support business processes. We characterise an information processing system, in general, by

1. a boundary, respectively scope,
2. a (functional) behaviour that supports a specific set of business processes, and
3. further non-functional properties.

We consider all levels of abstraction in the description of a system that have the purpose of supporting selected business processes, including the system’s architecture, used hardware, the underlying (network) infrastructures, and one or more applications. The term *application* exclusively covers aspects of software, independent of a particular system’s (technical) architecture.

RE as the Alignment of Information Systems and Business. The application domain of business information systems is steered by the principle of supporting a certain scope of an economically driven set of business processes with information systems [BCVP06, IIB09]. As described in Sect. 2.2.3.2, requirements can be specified at different levels of abstraction, including business requirements (see the abstraction hierarchy of Wiegiers [Wie03]), as well as system requirements (see

⁶ A detailed definition of the term *business process* is given in Chp. 4.

2.3 Fundamentals in Business Information Systems

the abstraction hierarchy of Gorschek et al. [GW06]). The alignment of information systems with the business needs thereby covers the philosophy of *requirements engineering* that aims at describing a problem space as comprehensively as possible, i.e., it aims at defining (system) requirements aligned to the needs of all relevant stakeholders (see [Wie03, FGP04, BPKR09], Sect. 2.2.1.1).

In Sect. 1.1 (problem statement), we argued, however, that RE is still not integrated into the views given in architecture frameworks for business information systems. Methods and techniques for the specification of requirements still are missing in the area of business information systems. On the other hand, available RE approaches still need a domain-specific interpretation (see Sect. 2.2.1.2). They emphasise the description of needs towards systems as a consequence of business needs. They do not include, for example, the engineering of business aspects, such as the re-design of existing business processes or the description of business processes with the intent of outsourcing processes and systems (or single parts of them).

Still, RE can be paraphrased with the specification and alignment of the needs towards the *business* with the needs towards underlying *information systems*. For a structured description of such needs, we make use of architecture reference models.

2.3.1.2 Architecture Reference Models

An architecture is the “organisational structure of a system or components” [IEE90]. A reference model is a model blueprint that conceptualises a chosen application domain in a standardised manner (see Sect. 2.5.1.1 introducing related terms).

Architecture reference models (or *architecture models*) give guidance to structure and describe an architecture in a standardised way. The term *architecture*, however, is frequently used in the domain of business information systems and considers different agreements. In the following, we introduce two use cases for the use of the term. The first use case exclusively considers the architecture of single systems (*system architecture models*). The second use case considers the term in a broader sense and refers also to business (processes) and their structuring as part of a business architecture. We refer to this with the term *enterprise architecture models*. Regarding enterprise architecture models, we describe two different approaches: *enterprise application integration* and *service-oriented architectures*. Each of the approaches considers the alignment of both the system architecture and the business architecture.

System Architecture Models

System architecture models reflect the basic concepts used to describe single system architectures. A system architecture is characterised by a specific structure with components, the relations in-between the components and to a system’s environment, and observable characteristics of the systems [IEE00, Cru09]⁷.

The observable characteristics of a system are described in terms of a usage layer [BFG⁺08], the highest level of abstraction for the description of a system. Usage layers involve user requirements, which describe how users intend to use the system when performing their business processes. Although the resulting functionality of a system (and its structure) is described according to the needs of, e.g., business processes, system architecture models do not offer means to describe

⁷ Although a system architecture is characterised in addition by architectural layers including a logical and a technical one, we do not describe them in detail, since the system architecture is not in scope of this thesis.

aspects that go beyond the usage of single, isolated systems. Hence, system architecture models emphasise single systems and their structure rather than their relations to other systems and to the business processes that they shall support.

The exclusive description of single systems leads for companies to the problem of historically grown monoliths with respect to their “system landscapes” – representing a set of interacting systems. Each system supports a chosen set of business processes and is described without taking into account a comprehensive view on further business processes and their relation to other systems. As a consequence, single systems have complex dependencies to surrounding systems. Changes in the business processes often lead to redundancies in the functionality, whereby changes not only affect single systems. The systems are not flexible w.r.t. changes, e.g., ones, which are performed if extending existing business processes. Therefore, system architecture models do not efficiently support the description of whole system landscapes [EHH⁺08].

Enterprise Architecture Models

Enterprise architecture models (also referred to as *organisation architecture models*) take a broader view on an architecture. Enterprise architecture models incorporate the structure of business processes and their alignment with systems. The description of systems additionally includes two different architecture perspectives: the information or information system architecture, including the description of applications, and the technology architecture with networks and platforms [Sch04]. The description of business processes is structured into logically related business processes, e.g., by means of business units. The structured business processes are baptised with the term *business architecture* (e.g., as done in the approach *Quasar Enterprise* [EHH⁺08]). The description of single systems is then performed according to the business architecture.

Structuring both business and systems according to each other aims at the alignment of IT and business. This alignment shall tackle the problem of redundancies given by system architecture models and, thus, shall achieve flexibility regarding changes (see also Schekkerman [Sch04]).

In the following, we briefly describe two concrete approaches: enterprise application integration and service-oriented architectures.

Enterprise Application Integration. *Enterprise application integration (EAI)* aims at giving guidance for the description and implementation of sets of dependent systems and applications. Although EAI tackles the problem of coupling single systems and, thus, achieve flexibility for changes in the business processes, it is a technology-driven concept.

EAI products make, for example, use of interfaces and adapters, which couple different systems. Hence, the alignment of systems and the business is not efficiently gained by EAI, since related approaches do not provide means going beyond a technical integration. EAI approaches emphasise the technical coupling of systems rather than their structuring according to the structure of the business processes of a company. Consequently, redundancies in the functionality of the systems are not effectively avoided when describing systems to support whole supply chains. Further information can be taken from [EHH⁺08].

Service-Oriented Architectures. *Service-oriented architectures (SOA)* extend the technical view of EAI approaches and structure the business architecture and the

2.3 Fundamentals in Business Information Systems

system architecture, both from a logical point of view. However, the notion of SOA comes with a variety of interpretations covering, inter alia, architecture principles and system design concepts, and methodological aspects for approaching the design of an architecture. Because of the different interpretations and existing controversies regarding SOA, we subsequently focus on those basic concepts to which our contribution relates. For this, we take into account the SOA manifesto [Gro09a] and our previous work [BLMF⁺10] in which we discuss the basic concepts of SOA from a theoretical point of view.

A SOA does, in general, not directly involve any technicalities, as known, for example, from the area of web services. Instead, a SOA is characterised by (business) domain structuring principles following long-term goals within a design methodology, including

- high compositionality and interoperability,
- high flexibility enabling changeability of system architectures respecting changes in the business (processes), and
- shared (system) services over specific-purpose implementations.

According to these principles, a SOA is meant to achieve flexibility in the design of an architecture with non-technical concepts. The structuring of system architectures is performed in a functional, business (process-)oriented manner in order to enable an efficient realisation of changes in business and system requirements.

A SOA aims at the flexible configuration of systems by the use of services that abstract from user-centric functionality of systems to support a chosen set of business processes. A SOA considers the alignment of both the business architecture and the system architecture(s) [Gro09b] by taking this logical and user-centric view.

In contrast to EAI, the alignment is supported by the description of an additional abstraction level in-between the business and the system levels, often referred as the *conceptual level* [EHH⁺08, Erl05, KBS04]. The alignment of both the business and the systems is performed within this additional level by means of services that give a logical representation of functionality the systems shall offer in order to support a selected set of business processes.

Although our artefact-based reference model is not specifically elaborated for SOA, we make use of two essential SOA principles:

1. We infer, according to a set of business processes, the functionality of systems from a user-centric view by means of, e.g., services. This principle is reflected in the basic concepts for modelling functionality among a hierarchy of levels of abstraction (see Sect. 4.2.2.1).
2. We follow the principle of *domain structuring* and structure business processes by means of, e.g., (business) domains, which group logically related business processes. This principle is reflected in the artefact model (including the concept for modelling business domains) and in the process model, e.g., by including concepts for arranging requirements elicitation workshops according to those domains (see also Sect. 4.8.2).

Further information can be taken from the artefact-based reference model in chapter 4.

Finally, how to describe an architecture according to a architecture model, depends on the chosen framework. We subsequently give an overview of existing frameworks for business information systems, which we use in Sect. 2.4 to give a first interpretation of RE for the application domain.

2.3.2 Architecture Frameworks for Business Information Systems

Architecture frameworks for business information systems offer methods, description techniques, and guidelines for the description of architectures. Most of the frameworks refer to enterprise architecture models and, thus, are known as enterprise architecture frameworks (see the foregoing section).

Available enterprise architecture frameworks can be categorised into several areas; for instance, approaches for describing architectures or approaches for assessing existing architectures. A detailed overview of the frameworks is given by Schekkerman [Sch04] and by the *Guide to the (Evolving) Enterprise Architecture Body of Knowledge* [Hag04]. In the following, we stick to frameworks for describing architectures.

Frameworks for describing architectures can be divided into activity-based ones and artefact-based ones. Activity-based frameworks constitute methodologies and give guidance on what has to be done. A prominent example is *The Open Group Architecture Framework (TOGAF)* [Gro06b]. Instead, artefact-based architecture frameworks give guidance by defining the artefacts that have to be produced and terminology to be used (in Sect. 3.1.3, we refer to this category of frameworks as *Ontology Architecture Models*).

One prominent artefact-based framework is the *Zachman Framework* [Zac87]. The Zachmann framework describes the artefacts that have to be produced in order to describe an architecture. These artefacts are defined as part of a matrix in which the rows horizontally define different “views”, such as the view of an owner, and the columns define different “focusses”, such as on the processed data. This matrix served as a basis for defining levels of abstraction in further frameworks which rely on the Zachmann framework.

One exemplary framework that makes use of such levels of abstraction is a commercial one, the *Integrated Architecture Framework (IAF)* [Gro06a] being developed by the Capgemini Group since the 1990s. IAF is based on experiences of architects, made in real life projects, whereby the framework represents best practices [Sch04]. We subsequently describe the levels of abstraction in the IAF.

2.3.2.1 Levels of Abstraction in existing Frameworks

Figure 2.5 illustrates, at the example of IAF [Gro06a], how levels of abstraction are incorporated by enterprise architecture frameworks. The figure is structured according to two dimensions. One dimension describes *aspect areas*, the other dimension describes *architectural layers*.

Aspect areas define the architectures that are in scope of the framework. The first area *Business* and the second one *Information* comprehend a view onto the business architecture (see the foregoing sections). The third area *Information System* and the fourth one *Technology Infrastructure* comprehend the IT infrastructure that is meant to support the business, respectively that is aligned with the business processes.

Architectural layers define vertical level abstraction among all the four architectures. Vertical abstraction is characterised with the *Contextual Layer* (“Why?”), the *Conceptual Layer* (“What?”), the *Logical Layer* (“How?”), and the *Physical Layer* (“With what?”). Compared to the architecture model introduced by Broy et al. [BFG⁺08], the last three introduced layers correspond to the usage layer, the logical layer, and the technical layer (see also the introduction into the levels of abstraction in Sect. 2.2.3.2). Each cell in the resulting matrix gives, with a question, guidance to architects (see also [EHH⁺08]).

2.4 Requirements Engineering for Business Information Systems

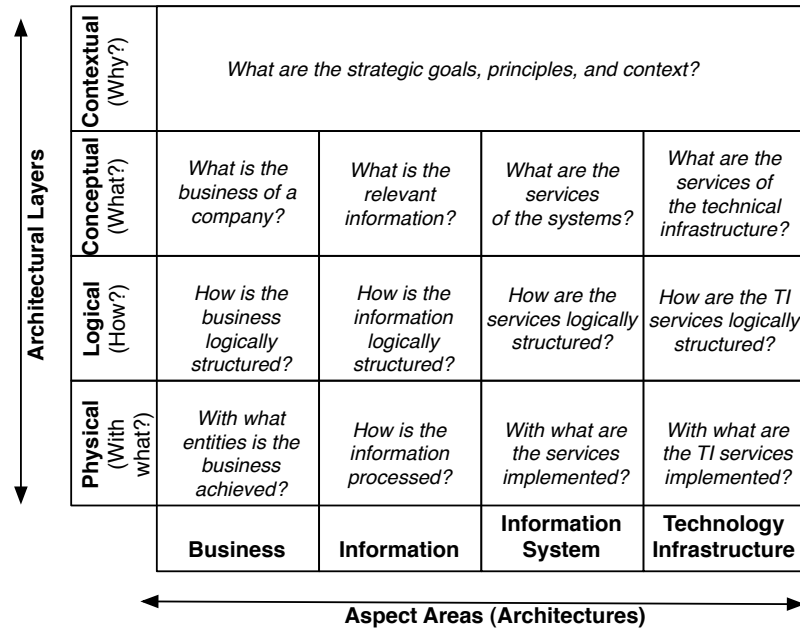


Figure 2.5: Levels of abstraction as defined in the *Capgemini Integrated Architecture Framework* [Gro06a].

In the context of the thesis, we take both the architectural layers and the aspect areas as orientation when defining the levels of abstraction in Sect. 4.2.

Finally, as stated in Sect. 1.1, aspects of requirements engineering are reduced in most frameworks to the contextual layers (“why shall something be done?”). As shown in Sect. 2.3.1.1, however, the principle of *IT business alignment* paraphrases at the same time characteristics of RE. Hence, we subsequently illustrate under this perspective to what extent IT business alignment and therefore business information systems analysis (as a domain-specific interpretation of RE) are to be covered by the framework.

2.4 Requirements Engineering for Business Information Systems

As shown in the problem statement in Sect. 1.1, there is a gap between the architecture-driven principles in the area of business information systems and the common agreement on the role of RE as a software engineering discipline. While available architecture frameworks for the application domain neglect the principles of RE to be an integrated discipline, available RE approaches still need a domain-specific interpretation (see also Sect. 2.2.1.2).

We discussed for this reason, as a first step, the principles of RE, which seek for an iterative approach of problem statement and problem solving (see Sect. 2.2.1.1). In the foregoing section, we then discussed the principles of the application domain under the perspective of IT business alignment. In Sect. 2.3.1.1, we showed that RE can be considered as this alignment with respect to

- aligning *business* needs, reflected, e.g., in business process models (business requirements) with

- the needs towards underlying *information systems*, reflected, e.g., in user and system requirements that describe expected user-visible system behaviour.

Taking this interpretation of the habitually used term *IT business alignment* as a first domain-specific interpretation of RE, we described in Sect. 2.3.2 an exemplary architecture framework for SOA introducing its levels of abstraction.

Figure 2.6 now illustrates, on the basis of those levels of abstraction, to what extent problem statement, problem solving, and the alignment of both is covered in the framework, i.e., to what extent RE should to be covered in the domain-specific, architecture-driven area of business information systems.

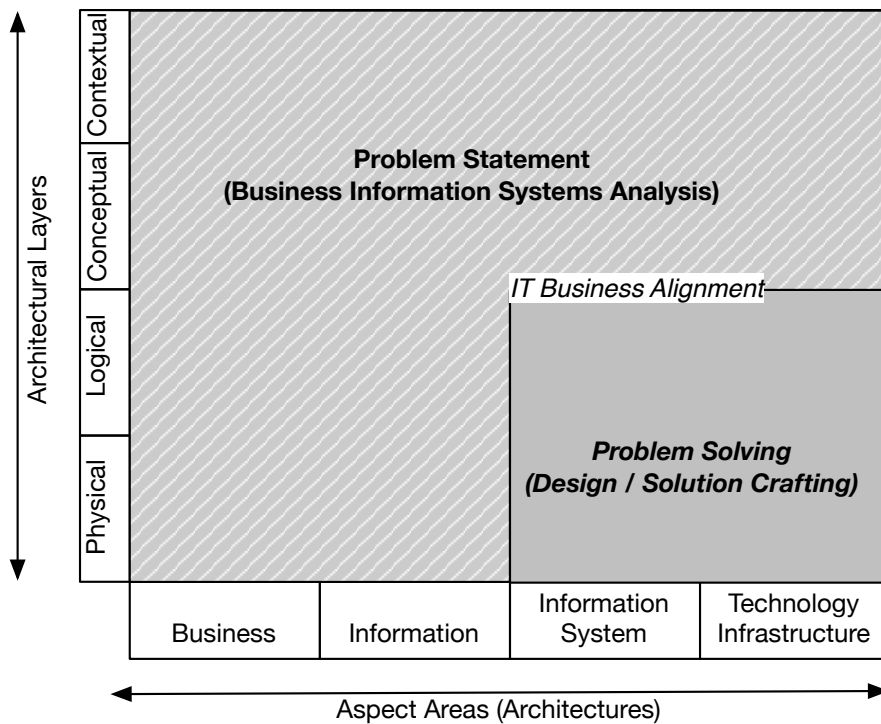


Figure 2.6: First interpretation of the extent to which requirements engineering artefacts relate to the domain-specific views in the Capgemini IAF (see also Sect. 4.1 introducing the discipline *BISA* on the basis of this first interpretation).

In the depicted framework, we divide the levels of abstraction into two major areas. The upper left area denotes the scope of RE within this exemplary framework. Note that it is, however, not possible to define any exact border line between problem statement and problem solving in a standardised manner, because

1. this depends on chosen methods and approaches for analysis and, thus, the architecture framework,
2. problem statement and problem solving is performed from a methodological point of view in iterations, strongly relating to each other, and because
3. “one man’s requirement is another man’s design” [Dav90], i.e., the term *problem statement* can be defined on different levels of abstraction in the chosen architecture framework depending on the view taken; for instance, the description of a logical architecture can be seen as a problem statement from the view taken in the design of a technical architecture.

However, in the context of this thesis, we propose the term *business information systems analysis* (BISA) as a rooftop over the principle of describing the needs of

2.5 Development Process Models

the business and resulting needs towards underlying systems taking into account the levels of abstraction in RE, which we introduced in Sect. 2.2.3.2. At the same time, we exclude any aspect that deals with designing an appropriate solution w.r.t. system's internals, i.e., the logical component architecture (given by a glass box view on a system).

Hence, we see RE to consider in the area of business information systems

1. any aspect that covers the business needs, including the business architecture [BCVP06, Wie03, GBB⁺06, DKOA06], and
2. any aspect that covers the needs and constraints towards systems [IIB09, GW06, Wie03, GBB⁺06], including also the transition to the (logical) systems architectures in terms of illustrated alignment, without explicitly specifying a possible internal realisation of a system's component architecture.

In Chp. 4.1, we further refine this first domain-specific interpretation of RE⁸.

2.5 Development Process Models

So far, we discussed the principles of RE, the ones of business information systems, and how both relate to each other, which we baptised with the term *Business Information Systems Analysis* (BISA). Since one aspect covered in our contributions is the process integration of our artefact-based reference model into a development process model, we now discuss the fundamentals and the related work in the area of development process models and meta models.

We define principles and used terms, and how the contributions of the thesis relate to those principles. We conclude this section with a discussion of related work.

2.5.1 Principles and Terms in Development Process Models

The construction of a development process model is performed at different levels of abstraction. One level of abstraction includes the definition of a *development process meta model*. A development process meta model describes what elements and relations have to be taken into account when designing a *development process model* for a particular application domain. The development process model, constructed according to the meta model, represents the resulting *reference model* to be used as orientation in development projects.

At each level of abstraction, we thereby make use of different terms and principles, which we introduce in the following. Based on the introduced terms, we describe how these relate to the contributions of the thesis.

2.5.1.1 Models and Reference Models

The term *model* considers two characteristics:

1. *Abstraction*: Models abstract from details of an object (that can be a model itself) and raise thereby “the level of abstraction at which the systems are developed” [BFH⁺10].
2. *Simplification*: When abstracting from an object by means of a model, we capture only those aspects that are of interest in order to achieve a specific purpose and thereby reduce complexity [Sta74, Fri06, FL03].

⁸ Section 3.3.3 additionally gives further information on how we determined the RE-relevant artefacts, respectively their contents, within the introduced levels of abstraction.

A model can thus be understood as an abstraction of an object while not capturing every detail of the object in order to reduce the complexity in the model (see also Kuhrmann in [Kuh08b]). In the area of model-based development, the use of models has become an indispensable support for specification and design, implementation, and verification and validation of software and systems. Since models can be used as a *reference* for a particular application domain, they can serve in addition as a basis for defining methodologies or terminology, both supporting a structured process and the communication within the process.

We see a *reference model* according to Winter et al. in [WS06] as a means to capture model blueprints for a certain application domain supporting the re-use of knowledge. We define the term for our context as follows:

Definition: Reference Model

A *Reference Model* defines concepts and relations to be used as orientation for a particular application domain.

Fettke et al. [FL03] discuss for this reason also the relation to further terms, like “model pattern”. An application domain, for which reference models are built, in turn, characterises the area in which a particular family of systems is used and, thus, impacts the followed principles when applying, for example, a methodology (see the foregoing section). The area of applying reference models is thereby manifold. One concrete area for using reference models is given by development process (reference) models, which we describe in the following Sect. 2.5.1.2.

Ontology “versus” Reference Model. Another term frequently used is *Ontology*. Ratiu defines in [Rat09] an ontology as follows: “An ontology represents a conceptual model of a domain in the form of named concepts arranged in a generalisation / specialisation hierarchy (taxonomy) and relations among them. The concepts and relations represent a consensual agreement on the domain by a category of domain experts.” Hence, an ontology is a reference model that defines terms and meanings of artefacts in relation to other artefacts. The artefact model defined in Chp. 4 represents in the nearer sense an ontology. In the area of artefact orientation, however, an ontology is only one possible conceptualisation of an artefact model. See also Chp. 3 where we lay the terminological foundation for the concepts used in the artefact-based approach.

2.5.1.2 Development Process Models

Development process models are reference models abstracting from a project-specific development process. According to Gnatz [Gna05], we define a development process model as follows:

Definition: Development Process Model

A *Development Process Model* is a standardised organisational reference model that abstracts from the idealised execution of a development project, including a description of artefacts (deliverables) to be produced, activities to be performed, and roles to be assigned.

Since this understanding covers different aspects with specific characteristics (artefacts, activities, roles), Kuhrmann introduces in [Kuh08b] a modular view onto

2.5 Development Process Models

development process models by means of *sub-models*. Figure 2.7 illustrates the different sub-models to be covered by a development process model according to Kuhrmann.

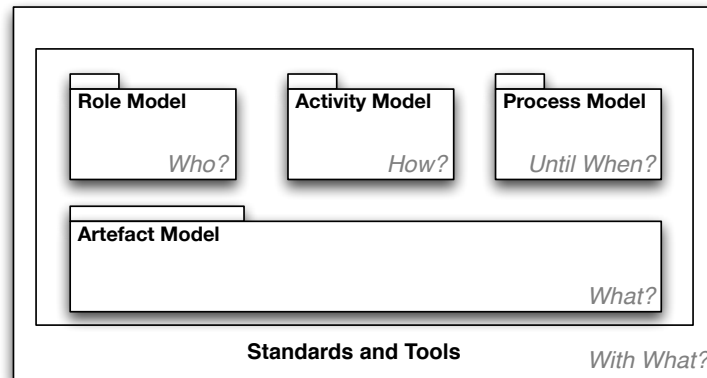


Figure 2.7: Sub-models of development process models according to Kuhrmann [Kuh08b].

A development process model consists of the following sub-models:

1. An *Artefact model* that describes what deliverables are produced or exchanged as an intermediate of a process or an activity (“what”).
2. A *Role model* that describes participants of a process with particular responsibilities and abilities (“who”).
3. An *Activity model* that describes what steps to perform in order to create, modify or use an artefact (“how”).
4. A *Process model* that arranges artefacts, roles, and activities as part of a concrete project execution by means of processes, phases, and milestones (“when”).
5. *Standards and tools* that conceptualise standards (including methods) and tools that can be used for a particular application domain (“with what”).

Each development process model can be, at least in parts, arranged by the introduced set of sub-models. The sub-models are not all obligatory while their structure and semantics also may differ. In fact, what sub-models to define and in what level of detail to define them depends on the followed philosophy. Method engineering, for example, puts emphasis on the definition of activities and methods for a chosen application domain (see also [BWHW05]). The other sub-models are then defined from this activity-centric view, e.g., by defining and coupling artefacts as an outcome of specific method in the activity model. If the development process model is defined following the artefact-based philosophy, the artefact model is in the primary centre of our attention. The other sub-models are then defined and coupled to the artefact model. In Chp. 3, we discuss the differences in detail.

In addition to the followed philosophy, an application domain also influences the definition of a development process model. There exist development process models defined for a particular application domain, and ones that are defined independent of an application domain. If constructing a development process model that is independent of an application domain, the modification and extension of the model for a particular application domain has to be performed on the basis of the defined structure and semantics of the development process model. This is, in turn, formulated as part of a *development process meta model*.

2.5.1.3 Meta Models and Development Process Meta Models

A meta model is often seen as a model “above” another model, i.e., a model that abstracts from another model. In the area of model-based development, meta models are used to raise one additional level of abstraction.

Furthermore, meta models are seen as an abstraction of (modelling) concepts of one particular domain and, thus, meta models can be also understood as reference models. Because a meta model, that describes the concepts to be used for a particular domain, can be abstracted itself, the different levels of abstraction (the “meta levels”) are not limited in their hierarchy (see also Hofstadter [Hof79]). Each meta level then defines constructs (concepts and relations) and rules, in the following named *structural properties*, to be used in the models that are situated in the directly underlying level. At the last two lowest levels of a meta hierarchy, the models do not take anymore an exclusive view onto structural properties. Instead, they take the view onto a particular application domain by defining the concepts and relations of the application domain as part of a *reference model*. Such a reference model then abstracts from the concepts used at the last level (see the next section for examples).

However, we aim at the definition of an approach that can be integrated into development process models. We thereby refer with *meta model* and *meta modelling* to the understanding and agreement given in the area of development process models.

Definition: Meta Model

A *Meta Model* defines a set of (reference) models.

We take a limited view onto a meta hierarchy and see a meta model from a more pragmatic view, in which a meta model captures the *abstract syntax* of a reference model by defining the constructs and rules used to construct a reference model.

Thus, a meta model is a model that exclusively defines the structural properties of a model and, thus, formulates the language to be used for the description of a model [Gna05, Kuh08b]. A development process meta model consequently formulates the language to be used for the description of development process models.

In the following section, we use the introduced terms to define the setting of our contributions with respect to the multi-layered meta hierarchy of the *Meta Object Facility* (MOF) [OMG06b] and with respect to the understanding in the area of development process models.

2.5.2 Setting of Contributions in Context of Development Process Models

In this section, we define the setting of our contributions with respect to the different levels of abstraction in which development process models can be described.

Table 2.2 illustrates for this purpose four levels of abstraction, as they are used in MOF (M3 to M0). MOF is an approach of the *Object Management Group* (OMG) for the description of modelling languages in which the meta models define the constructs to be used for defining languages. Based on the discussion of Kuhrmann in [Kuh08b], we subsequently describe the different levels of abstraction, how they are represented with the *Unified Modeling Language* (UML) [OMG10a, OMG10b], and how these levels are typically referred in development process (meta) models according to Henderson-Sellers et al. [GPHS06]. Because of the closeness of our

2.5 Development Process Models

Table 2.2: Meta levels in the context of contributions of the thesis.

	Hierarchy used in MOF	View taken by UML	View taken by Development Process Models	View taken by Contributions of the Thesis
M3	Meta Meta Model	MOF	–	–
M2	Meta Model	UML Language(s)	Development Process Meta Model (e.g., SPEM)	Meta Model for Artefact Orientation (Chp. 3)
M1	(Reference) Model	Class Model	Development Process Model (e.g., RUP)	Artefact-based Reference Model (Chp. 4)
M0	Instances	System at Run-Time	(Enacted) Project Model / Project Plan	Project-specific Exemplars (Outcome of Customisation at Project Level (Chp. 5))

contributions to the last of the mentioned areas, we take the third view as orientation for defining afterwards the setting of our contributions.

The depicted hierarchy considers the following four levels of abstraction:

- M3 describes the “meta meta model” to capture the language for describing meta models. The Meta Object Facility (MOF) itself is situated at this level offering a basis for the definition of UML meta models.
- M2 describes the “meta model” to capture a model of elements and relations found in modelling languages. The UML languages are described by the OMG at this level with the *UML Infrastructure* [OMG10a] and *Superstructure Specification* [OMG10b].
- M1 describes a concrete “model” as a reference that abstracts from a specified system at run-time.
- M0 finally represents that system at run-time.

By taking this view, we consider with M2 the abstract syntax of a concrete modelling language (like UML activity diagrams). We consider with M1 a project-specific modelling viewpoint from which code can be generated at M0 representing, in turn, the system at runtime.

By taking, however, the view from development process models, as they are discussed by Henderson-Sellers et al. [GPHS06], we consider only M2 to M0 as relevant while giving for each level a more abstract interpretation as done in the context of the UML.

View taken by Development Process Models. In the following, we give a brief summary of the levels of abstraction as they are understood in the area of development process models. We consider the levels M2 to M0 as the necessary ones to capture the concepts for the description of development processes.

M2 defines the abstract syntax of a development process model. Thus, a development process meta model occupies M2 and defines at this level the constructs, which are necessary to define the sub-models of a concrete development process model (such as the concepts “Role” or “Artefact”). An exemplary approach that can be allocated to this level is the *Software & Systems Process Engineering Meta-Model* (SPEM), which is under management of the OMG. Although most development process meta models do not instantiate a particular meta meta model,

SPEM is an exception in this context and refers to MOF as its meta meta model (see [OMG08], page 20).

M1 then defines an instance of the meta model, i.e., an interpretation of each sub-model defined by the meta model. This interpretation serves as a project-specific reference model; for instance, as given with the *Rational Unified Process* (RUP). A development process model defines, for example, instances of the roles concepts including, e.g., a “Requirements Engineer” or instances of the artefacts concepts, e.g., a “Requirements Specification”.

M0 then defines a comprehensive process viewpoint as an outcome of *enacting* the development process model, i.e., as an outcome of applying the development process model to a particular project [GPHS06]. This level includes project-specific exemplars of the artefacts (like the “Requirements Specification: Travel Ordering System”) or concrete roles (“Requirements Engineer: Mr. Sellers”), which perform activities and methods for creating the artefacts. These project-specific exemplars can be defined as part of a project plan [Kuh08b]. The transition from M1 to M0 (the *instantiation*, respectively the generation of exemplars) is usually performed via a static customisation approach, which is topic of our discussions in Sect. 2.6.

View taken by the Contributions of the Thesis. According to the introduced views, we allocate the contributions of the thesis to the levels of abstraction (see also the right column in Tab. 2.2).

The *Meta Model for Artefact Orientation* occupies M2 and is defined in Chp. 3. This meta model describes the abstract syntax of our artefact-based reference model for the domain of business information systems analysis. We describe with the meta model the concepts and relations of the reference model and the procedure performed during the instantiation. In the following, we refer with the term *meta model* exclusively to this level of abstraction. Note that the meta model presented in Chp. 3 does not explicitly refer to a particular meta meta model (as SPEM), although we refer to many concepts defined in MOF (e.g., typed associations).

The *Artefact-based Reference Model for Business Information Systems Analysis* (short: reference model or BISA reference model) is situated at M1 and defined in Chp. 4. With the reference model, we describe the domain-specific instance of the meta model, i.e., the reference model to be used, at project level, as orientation for the domain of business information systems.

The application of the BISA reference model to a concrete project (M0) is described as part of the *Artefact-based Customisation Approach* in Chp. 5. The customisation approach defines how to create *Project-specific Exemplars* of the reference model at M1 for a particular project at M0 in dependency to chosen project characteristics.

Further Concepts used in the Context of the Thesis. For the description of the meta model for artefact orientation (Chp. 3) and for the description of the artefact model (Chp. 4), we make use of UML class diagrams [OMG10a, OMG10b].

2.5.3 Related Work in Development Process Meta Models

There exist several contributions in the area of development process models, which are based on a meta model; for example, the *V-Modell XT* [FHKS08, VMX] offers an own meta model [TK09] or the *Rational Unified Process* (RUP) [KK03, JBR99] is based on the *Software & Systems Process Engineering Meta-Model* (SPEM) [OMG08] from the OMG. On the other hand, there are development process models for which there exists no meta model. This is mostly the case for approaches that

2.5 Development Process Models

are specifically elaborated for a particular application domain. For instance, the *SOAM* approach in the area of SOA takes into account the sub-models of a development process model introduced in Sect. 2.5.1.2, but has no underlying meta model.

However, it is necessary to define the abstract syntax of the development process model for the purpose of customisation. Otherwise, customising a development process model would rather rely on the expertise of the process engineer and project participants.

In the following section, we give an overview of related work in the area of development process meta models. In a second step, we introduce the V-Modell XT.

2.5.3.1 Overview of Development Process Meta Models

We take into account three major sets of contributions: SPEM and SPEM-based approaches; OPF and OPF-based approaches; the V-Modell XT. The first two sets of approaches can be allocated to the area of activity orientation, the V-Modell XT to the area of artefact orientation.

In the following, we give an overview of the first two sets of approaches, before introducing the meta model of the V-Modell XT in detail. Further information are also given in Sect. 1.3, where we discuss the limitations of activity-based approaches regarding customisation capabilities and in Sect. 3.1.1 where we discuss activity-based approaches in comparison to artefact-based ones.

The *Software Process Engineering Meta-Model* (SPEM) [OMG08] is a development process meta model of the OMG. It offers a language for the description of any kind of software engineering processes. SPEM is based on MOF [OMG06b] and reuses further concepts found in OMG specifications such as the UML infrastructure [OMG10b]. Hence, we can refer to several diagram types of the UML for the description of SPEM and SPEM-based approaches. The operationalisation of SPEM in a tool-supported manner is given by the *Eclipse Process Framework* (EPF) [Hau06a, Hau06b]⁹. EPF is an Eclipse-plugin that offers a framework for the composition of processes according to the meta model. Figure 2.8 illustrates the package structure of the meta model SPEM.

The *Core* package includes shared classes and abstractions from classes found in the process structure. The *Process Structure* itself contains the elements to describe activities (and hierarchies of activities), roles, and work products. The *process behaviour* package represents the interface to behaviour modelling, i.e., links to further behavioural diagram types of the UML, such as activity diagrams, used to describe the work flow of a process. The *Managed Content* package includes concepts for describing (textually) the process documentation. It includes, for example, best practices that are related to the process structure. The *Method Content* package includes best practice-related concepts for documenting methods and description techniques. The *Process with Methods* package integrates these methods into the process structure by means of, for example, milestones. Finally, the *Method Plugin* package includes concepts for defining configurations of methods on the basis of an individually defined method library.

SPEM follows, as already mentioned, an activity-based philosophy. This philosophy is reflected in the package structure, which emphasises processes and methods. On the basis of SPEM, several development process models are defined, such as the *Rational Unified Process* (RUP) [KK03, JBR99] or the *Open Unified Process* (OpenUP), which both incorporate the same activity-based philosophy as SPEM.

⁹ See also <http://www.eclipse.org/epf/>, accessed in July 2010.

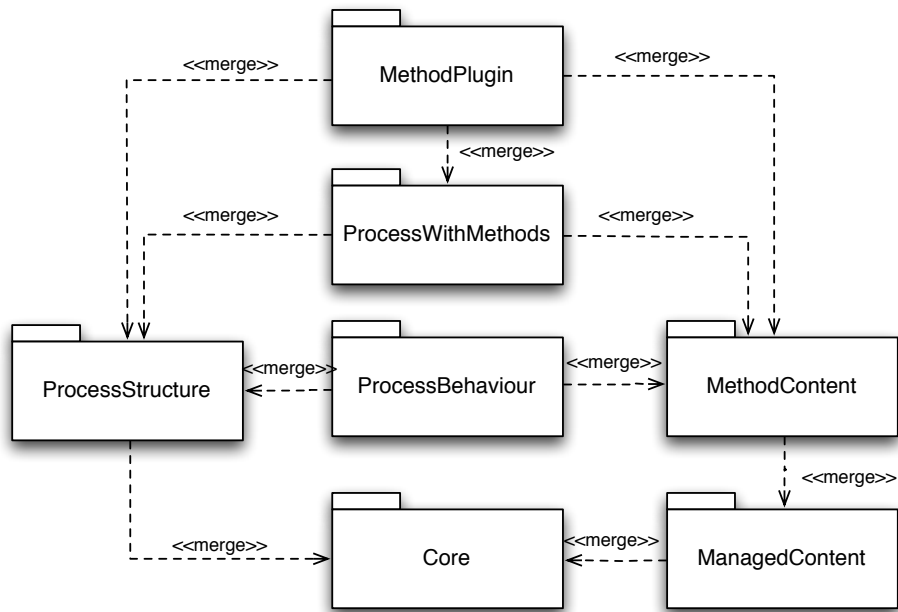


Figure 2.8: Packages of the *Software Process Engineering Meta-Model* (SPEM) [OMG08].

A similar philosophy can be found in the *OPEN Process Framework* (OPF) [GHSY97], a further framework for the description of development process models. Although constituting a detailed meta model, OPF is not logically structured according to the sub-models of a development process model (as shown in Fig. 2.8). In addition, OPF has not been maintained since 2005 [Kuh08b]. It is, however, a recognised source for the *International Organisation for Standardisation* (ISO) to establish the *ISO/IEC 24744:2007 Software Engineering – Metamodel for Development Methodologies* (SEMDM). A recognisable characteristic of SEMDM is the hierarchical structure of not only activities, but also of artefacts reflecting, e.g., hierarchically structured documents. Since SEMDM is based on the principles of method engineering, it still incorporates the same activity-based philosophy as SPEM.

2.5.3.2 The V-Modell XT Meta Model

The V-Modell XT is a process framework, obligatory for all governmental (IT) development projects in Germany. It was developed as a response to the growing demand towards a flexible development process model and, thus, displaced the activity-based V-Modell 97 [DW99]. The V-Modell XT offers a development process model [FHK08, VMX], a meta model [TK09], a concept for customisation [KTF10, KH08], and a set of tools incorporating the philosophy of artefact orientation [Kuh08a].

The meta model of the V-Modell XT [TK09], in its release 1.3, describes the abstract syntax and is logically structured into five packages (see Fig. 2.9).

The package *Basis* includes common elements that are shared by the static and dynamic elements, such as the process documentation. The package *Static Elements* includes the sub-models necessary to describe the concrete process, like a role model or an artefact model. The process is described with the concepts in the package *Dynamic Elements*, such as with the process model or with project execu-

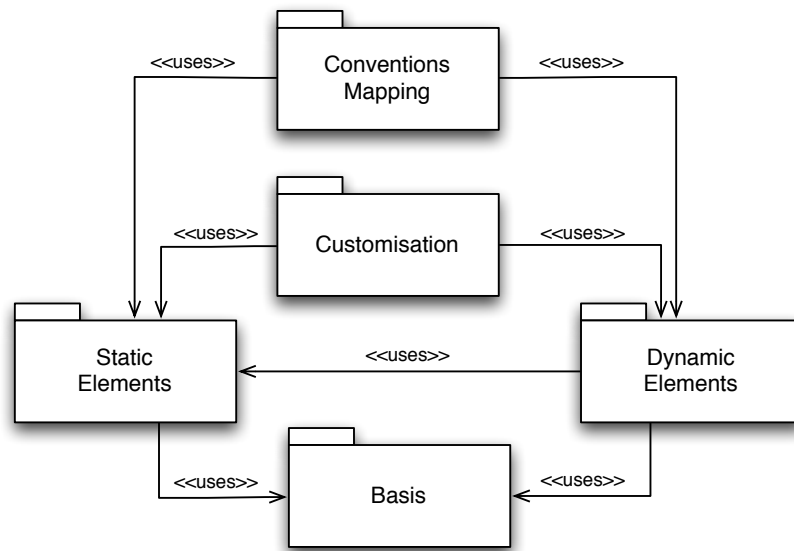


Figure 2.9: Packages of the V-Modell XT meta model defined in [TK09].

tion strategies. The package *Customisation* includes elements of a project characterisation and the customisation of the dynamic and static elements according to chosen project characteristics (see also Sect. 2.6.4.2). Finally, the package *Conventions Mapping* includes the mapping of terms and concepts used in the V-Modell XT in relation to, for example, other development process models.

In the following, we describe the parts of the meta model that are in scope of the process integration in Chp. 6. Detailed information about the overall meta model can be taken from [TK09, Kuh08b, Fri06].

Static Elements: Product Types

The package *Static Elements* contains all sub-models of a development process model that are used to define, structure, and describe the process. It contains, for example, artefacts, activities, and roles. This sub-model is used for generating the process documentation using the content description of the particular model elements.

Product Types. The artefacts define the potential results of a development process, in the V-Modell XT named *Product Types*. Figure 2.10 illustrates the concepts used for the product types.

Product types are structured into *Topics* to be considered. The structuring of topics into *Sub Topics* leads in the V-Modell XT to the hierarchical structure of the product types aiming at a structured customisation¹⁰. The V-Modell XT provides, in addition, with *Disciplines* containers to combine product types.

Example 2.1 illustrates an excerpt of the concrete artefacts and corresponding disciplines, which are structured according to management issues and according to development issues.

¹⁰ Chapter 3 gives further information about perspectives that can be taken onto artefact models depending on their purpose, e.g., the purpose of achieving flexibility in the context of development process models.

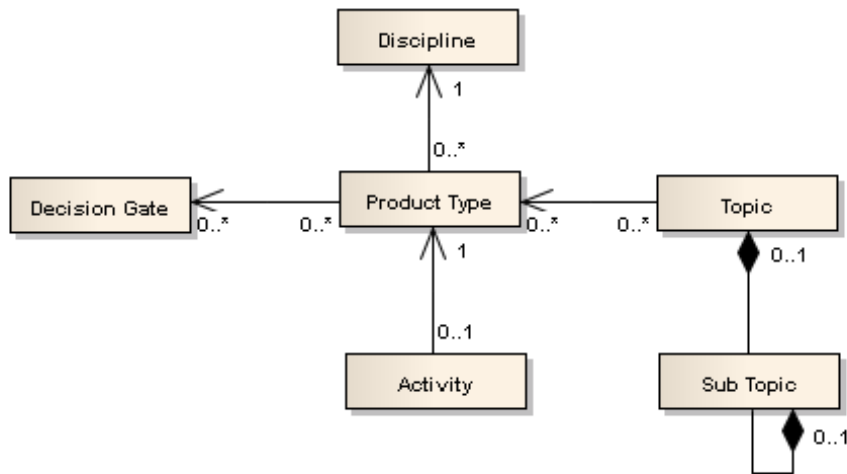


Figure 2.10: Static view on product types in the V-Modell XT meta model.

Example 2.1. *Set of Product Types and Disciplines in the V-Modell XT*

Exemplary Management Product Types		Exemplary Development Product Types
Planning and Control <div> <div>E</div>Project Progress Decision </div> <div>Project Manual</div> <div>I</div> QA Manual <div>Project Management Infrastructure</div> <div>Estimation</div> <div>Risk List</div> <div>I</div> Project Plan <div>Work Order</div> <div>Life Cycle Cost Calculation</div>		Reporting <div>Meeting Document</div> <div>E</div> Project Status Report (Supplier) <div>E</div> Final Project Report (Supplier) <div>Project Diary</div> <div>Measurement Data</div> <div>Metrics Analysis</div> <div>Commercial Project Status Report</div> <div>Project Status Report</div> <div>Quality Status Report</div> <div>Final Project Report</div>
		Requirements and Analysis <div>User Tasks Analysis</div> <div>Hazard, System Safety and Security Analysis</div> <div>I E</div> Project Proposal <div>I</div> Requirements Specification <div>I</div> Requirements Evaluation <div>Legacy System Analysis</div> <div>Market Survey for Off-the-Shelf Products (Acquirer)</div> <div>Market Survey for Off-the-Shelf Products (Supplier)</div> <div>Make-or-Buy Decision</div> <div>I E</div> Proposal for the Introduction and Maintenance of an Organization-Specific Process Model <div>I</div> Requirements Specification Overall Project <div>I</div> Evaluation of the Overall Project Requirements Specification

Disciplines additionally contain *Activites*, which are used to describe the creation of products including, in part, a reference to several possible methods. Since the activities (being unconnected to each other) allow no description of a concrete work flow, the V-Modell XT provides means to order the creation of the products by assigning product types to *Decision Gates*. Decision gates then become after instantiation milestones in a project plan. If a decision gate shall be reached, the set of assigned product types has to be finished, including successful quality assurance. Decision gates are the interface to the dynamic elements that define an ordered sequence of decision gates, which in combination, lead to a concrete *Project Execution Strategy* (see also the next sections).

Product Dependencies. The meta model provides different dependency types for the products. These dependencies support, for example, the creation of product types w.r.t. the content of the product types. Figure 2.11 illustrates the dependencies that are in scope of the thesis: the *content-related* and the *creational* product dependencies.

2.5 Development Process Models

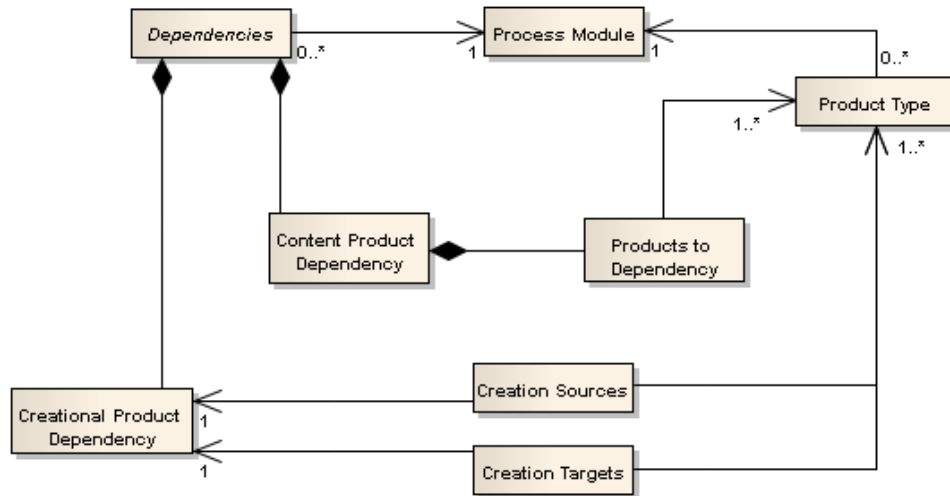


Figure 2.11: Product type dependencies in the V-Modell XT meta model.

Creational dependencies are used to describe why a particular product type has to be created in a project (the products' rationale being another product type). The sources of the directed dependency are the *Initial Product Type*; for instance, if considering a project plan, a status report has to be created, too. In example 2.1, the project plan is marked for this reason with an "I" (initial).

Content-related dependencies connect product types with each other and express that contents of one created product depend on the contents of another one; for instance, the system under consideration relies on the corresponding system specification.

Information on further dependency types can be taken from [Kuh08b].

Further Associations. While product types depend on other product types, other elements of the development process model (like roles) have to be taken into account, too. Figure 2.12 illustrates the associations between product types and decision gates. The same structure of the depicted association is used to connect product types with other elements, such as roles.

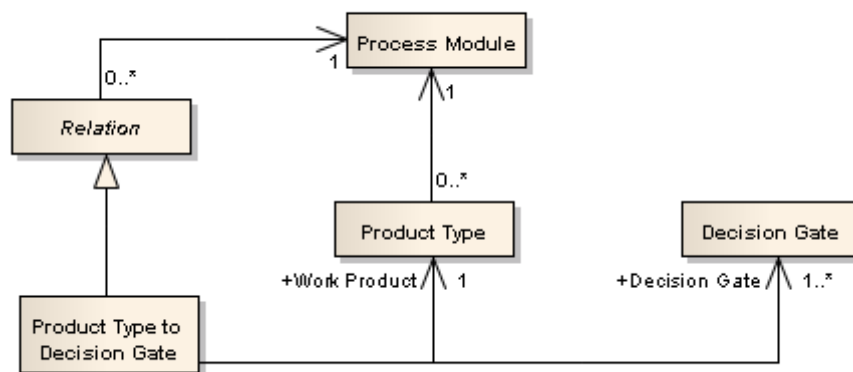


Figure 2.12: Associations of product types to further elements in the V-Modell XT meta model.

Dynamic Elements: Project Execution Strategies

The static elements of the meta model define the structure of the process. For the definition of a concrete process, the V-Modell XT refers to *Project Execution Strategies*. Project execution strategies state which decision gates have to be reached in which order. Since all product types are assigned to decision gates, the dates for the finalisation of the products are defined (within the project plan).

A further concept, used to define a project execution strategy, is the *Procedure Module*. A procedure model defines a partial process by a subset of decision gates in an given order. In a concrete project, several procedure modules are used to build a concrete project execution strategy in dependency to chosen project characteristics. Further information is given, however, in Sect. 2.6 (introducing the fundamentals and related work in the area of customisation).

2.5.3.3 Relation to Contributions of the Thesis

The the artefact-based reference model for business information systems analysis (Chp. 4) and the artefact-based customisation approach (Chp. 5) are based on a similar artefact-based philosophy that is reflected in the V-Modell XT (and its meta model). In particular, we define an artefact model as the backbone of project execution. We allocate the necessary process elements (activities, milestones, roles) to artefacts and establish a concrete process by the selection, arrangement, and creation of artefacts that are in scope of a project. In contrast to the V-Modell XT, however, we also consider the dynamic specification of the artefacts w.r.t. their domain-specific contents (see also Chp. 3).

2.6 Customisation of Development Process Models

In this section, we analyse the principles and related work in customisation of development process models. As a first step, we discuss different stages in which a development process model can be customised and the relation of our contributions to those stages. Afterwards, we classify related work, before discussing available approaches according to two particular areas: domain- and discipline-specific contributions, and contributions in development process models. We conclude with a summary and a discussion of related work.

2.6.1 Customisation Stages and Relation to Thesis

The customisation of a development process model can affect the different sub-models (activities, methods, artefacts) and be performed at different levels of abstraction (see also Sect. 2.5.2 introducing the hierarchy). According to [MFK09, KH08, Kuh08b], we distinguish, in general, between three different stages in the customisation of a development process model.

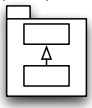
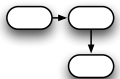
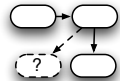
Table 2.3 illustrates these stages and depicts for each of the stage frequently used terms, a short description, and how our artefact-based customisation approach in Chp. 5 relates to the stages.

We subsequently introduce the three customisation stages, before discussing related work. We organise the stages according to

1. *Customisation at Organisational Level* considered by stage 1, and
2. *Customisation at Project Level* considered stage 2 and 3.

2.6 Customisation of Development Process Models

Table 2.3: Overview of customisation stages and terms in the context of the thesis.

	Envisioned Level of Abstraction	Frequently used Terms	Description	View taken by Artefact-based Customisation
Stage 1	Development Process (Meta) Model 	- Life Cycle Management - Structural Tailoring	- Creation and Maintenance of Development Process Model according to Release Plan	-
		- Process Integration	- Integration of Approaches into Development Process Model	Process Integration (Chp. 5.2)
Stage 2	Project Model / Project Plan 	- Static Tailoring / Customisation - Enactment	- Planning of Project and Instantiation of Project Model as, e.g., Project Plan according to Project Parameters	Initial Project Set-Up (Chp. 5.3)
		- Dynamic Tailoring / Customisation - Content Creation	- Execution of Project according to Project Plan and dynamic Reaction to (unplanned) Circumstances (Project Parameters)	Project-specific BISA Execution Strategy (Chp. 5.3)
Stage 3	Project Execution 			

2.6.1.1 Customisation at Organisational Level

Customisation at organisational level considers, in general, the creation, the administration, and the integration of a development process model [Kuh08b] including its modification (extension, reduction and / or specialisation of the sub-models' content [KH08, KTF10]). Customising a development process model at organisational level is, however, twofold.

On the one hand, the customisation can consider the management of the overall life cycle of a development process model, including

1. the inference and administration of different variants from a meta model for, e.g., defining organisation-specific reference models [Kuh08b]. These organisation-specific models can be extended, reduced, and specialised by enriching chosen elements with further concepts [KH08].
2. the direct modification of the meta model as part of a new release, e.g., by establishing new association types [Kuh08b].

The life cycle management describes either the inference and modification of a development process model according to the meta model, or it describes the modification of a meta model itself. Hence, this kind of customisation is also known as *Structural Tailoring*.

On the other hand, we can exclusively consider a *Process Integration* as we do in our contribution (see Chp. 5.2). A process integration can consider the integration of new concepts (single methods, roles, milestones, or artefacts) into a development process model [KH08, MFK09], or the integration of one development process model (or an excerpt of the like) into another with respect to given meta models [Kuh07].

Hence, we intentionally use the term *process integration* in a broader sense:

Definition: *Process Integration*

A *Process Integration* considers the integration of approaches into a development process model by establishing the associations between the concepts of the approaches to be integrated and the concepts of the development process model.

A prerequisite for a process integration is that the models to be integrated have an underlying meta model and that the elements and relationships in both meta models can unambiguously be mapped onto each other.

In Sect. 6.2, we perform a process integration of the artefact-based reference model of Chp. 4 into the V-Modell XT. Besides the process integration, we perform also the inference of the artefact-based reference model at M1 (Chp. 4) according to the meta model at M2 (Chp. 3). Further information on the inference is given in Sect. 3.3.3.

2.6.1.2 Customisation at Project Level

The customisation at project level brings a development process model to life by adapting it for its execution in a particular project according to individual project parameters. We see a project parameter as an assessable condition from inside or outside the project that influences its execution (def. in Sect. 5.3.2.1).

A customisation at project level consists of two stages: the first stage describes the approach of initially setting up a development project according to the development process model; the second stage describes its continuous customisation during project execution. Both stages are described in the following.

Static Customisation. The first step in the customisation of a development process model for a particular project describes the transition from M1 to M0 according to known project parameters. Most of related work that refers to the term *tailoring* is situated at this stage. A frequently used term that suitably characterises this procedure is *enactment*.

Enactment describes, according to Gonzalez-Perez et al., the procedure of “applying a methodology to a particular project”, while the project entities and their relations being created are governed by the methodology [GPHS06]. Hence, how the enactment is performed depends on the followed philosophy, i.e., the definition of the sub-models of a development process model by means of a meta model. Enacting a process model can be performed by selecting a specific sequence of activities to be executed (directly defining a process), if following an activity-based development process model; it can also be performed by selecting a subset of artefacts and building exemplars in a particular order of creation, if following an artefact-based philosophy (see also Sect. 3.1.1 discussing both philosophies).

The enactment of a process considers, in general, the creation of exemplars of the elements and relations described by the development process sub-models and, thus, the creation of a project model (or a project plan in case of “plan-based enactment”), which is based on the defined structure defined in the development process meta model. A frequently used term is also *static tailoring*. The term *tailoring*, however, can be understood as “cutting off specific pieces of a model” (like trimming or pruning) what is not exclusively the case. For this reason, we also depict in Tab. 2.3 the more generic term *static customisation*.

In the thesis at hand, we describe the approach at this customisation stage as the *Initial Project Set-Up*, which we introduce in Chp. 5.3.

Dynamic Customisation. Once a project is enacted, it is executed following, e.g., the project plan. This project execution considers the performance of the selected methods and / or the creation of the artefacts, whereby this stage can be paraphrased with the term *content creation* [KH08, Kuh08b].

2.6 Customisation of Development Process Models

Since it is in the nature of development projects that unplanned circumstances cause projects to deviate from the initially planned execution (see also Parnas et al. [PC86]), this stage has also to consider the reaction to such deviations. Although there still exists little guidance in this area, the reaction is known as *dynamic tailoring*, respectively *dynamic customisation*.

Dynamic customisation considers the adaptation of the enacted project entities during the project execution according to project parameters, which arise during the execution. Dynamic customisation can be performed during the whole development life cycle affecting the project plan, but also within a particular project phase, in which project participants have to reflect on arising project parameters and act according to those parameters within the pre-defined (enacted) setting.

In the thesis, we refer to the second variant, since RE is characterised due to the closeness to the customers' domain by uncertainty [EW05] so that many project parameters are not clear from the beginning of a project [BPKR09]. The procedure for this dynamic customisation is introduced with the *Project-specific BISA Execution Strategy* in Chp. 5.3. We describe within a planned project setting the dynamic content creation of the artefacts in the necessary and possible degree of completeness. However, we do not take a plan generation itself into account (further information is given in Sect. 5.1).

2.6.2 Classification of Related Work

In the following, we classify related work to perform a process integration and for a customisation of development process models at project level. In subsequent sections 2.6.3 and 2.6.4, we describe for the second area related approaches in detail.

2.6.2.1 Classification of Related Work for a Process Integration

Considering the modification of development process models at organisational level, there exists related work in the field of process management and process optimisation, such as the *Capability Maturity Model* (CMM) [PWBea94] and CMM-based approaches, which are specifically elaborated for chosen disciplines (e.g., for RE [BHR05]). Recent work has also been done in the definition of business cases and best practices for the introduction of RE to companies as a general topic [EW05].

Regarding the area of process integration, there exist approaches, which cover the integration of methods (see also Hammerschall [Ham08]). *Method integration* and *method engineering* in general, however, cover only one aspect of a process integration and disregard the other development process sub-models to be integrated into a development process model. Further work has also been done in context of the V-Modell XT w.r.t. its integration into an organisation via four stages [KTF10]: Analysis of the current situation and identification of needs in the organisation; Conceptualisation of the development process model (e.g., definition of organisation-specific roles); Realisation; Integration into the organisation.

However, the process integration of a domain-specific approach into a development process model eventually remains a topic to be covered by the approach to be integrated. Hence, there exists no directly related work in the area of process integration. Still, we use the four generic stages proposed for the V-Modell XT as orientation for constructing our process integration approach in Chp. 5.2.

2.6.2.2 Classification of Related Work for Customisation of Development Process Models at Project Level

The topic of customising development process models according to the needs of individual project environments has received much attention for many years, especially in the area of activity-based approaches. Available contributions evolve from early work of, e.g., Basili et al. [BR87]. It is recognised, however, that giving guidance for customising activity-based models has its limitations.

The *Tailorable Process for Software Engineering* [CCR⁺95] or the *V-Modell* 97 [DW99] give evidence on that fact, since the included customisation approaches are

- either defined as a highly complex task in which one has to select a sequence of activities according to a pre-defined set of project parameters while both are hardly extensible, or they are
- generically described and thereby not applicable.

This problem can, in general, be found in development process models, which rely on the activity-based philosophy (see also Sect. 1.3). For instance, the SPEM documentation [OMG08] lacks for concepts for enactment; these are informally circumscribed on 11 of 236 pages and give no concrete guidance while leaving this topic to the expertise of project participants or tool vendors. In fact, available surveys, such as the one of Predreira et al. [PPLB07], point out the general absence of practice-based research and experiences that would give evidence on the applicability of such customisation (respectively enactment) approaches. Most of available case studies on the application of activity-based development process models give attention to the resulting process rather than to the actually performed customisation approach.

For this reason, we supplement activity-based development process models with another area: *Situational Method Engineering*. This sub-area of method engineering contributes extensible approaches to the selection and integration of methods according to project parameters, without giving much attention to particular development process models.

Nevertheless, the area of activity-based development process models and method engineering contribute customisation mechanisms rather than concrete guidelines, which, e.g., take into account particular project situations. Thus, we additionally have to consider that the steps performed during customisation also depend on project-specific contents [WK92, THV97].

Therefore, we also investigate the area of decision support systems and, in particular, the area of *Content-Centric Decision Support Systems* (see also [RPA⁺01, HFIM06]). We perform this investigation according to Ngo-The et al. [NTR05] that contribute a literature survey on decision support systems. They define decision support as an approach to select, classify, and rate a set of alternatives in the performance of a project. The alternatives are based, in turn, on project-specific contents, such as requirements. The area of content-centric decision support systems makes use of multi-project environments that assist the selection of domain-specific description techniques according to project parameters.

Available literature contributes in surveys on decision problems and project parameters, and it contributes on the selection of specific description techniques according to those project parameters. The area additionally can be classified according to specific disciplines, like RE, including domain-independent techniques and domain-specific ones.

In summary, we classify related work into two areas, as illustrated in Fig. 2.13. The first area to be investigated constitutes domain- and discipline-specific contributions, which provide means to characterise projects with respect to a particu-

2.6 Customisation of Development Process Models

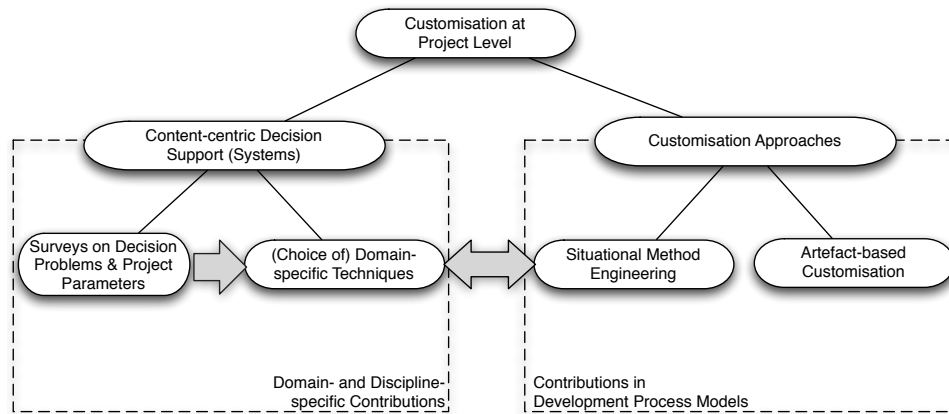


Figure 2.13: Classification of related work considering customisation at project level.

lar discipline and application domain. Surveys on decision problems and project parameters directly contribute to approaches for selecting description techniques. These approaches, in turn, complement the area of situational method engineering that belongs to the second area: customisation approaches. Situational method engineering is investigated besides artefact-based approaches as part of the contributions in development process models. These describe, in contrast to the first area, rules and mechanisms for performing the customisation.

Static versus Dynamic Customisation in Related Work. The introduced classification of related work makes no differentiation on whether approaches consider static or dynamic customisation. In fact, only few approaches make a statement about the customisation stage they address. A known exception is the V-Modell XT that explicitly considers the initial project set-up (stage 2).

Hence, we allocate available approaches to the static customisation stage. This is especially true for approaches, which belong to situational method engineering, because

1. the problem of compatibility of different methods (regarding syntactic consistency) is an issue that still is not completely solved [Ham08, THV97, Bri96]. This problem would be additionally hardened if having to weave methods into a predefined activity path during project execution.
2. project parameters that are taken into account remain trivial and / or coarse grained [THV97] as given by the V-Modell XT that intentionally describes these parameters in a generic way to be able to perform the customisation during initial project set-up.

2.6.3 Domain- and Discipline-specific Contributions

Domain- and discipline-specific contributions are specific-purpose approaches with a limited scope. Most of the approaches propose specific solutions, i.e., description techniques and / or methods, in combination with a specific problem, i.e., project parameters that argue for the proposed solution [NTR05]. These solutions are developed

- either for a specific purpose (a concrete project situation), such as:
 - given by description techniques, independent of an application domain. For instance, Doerr et al. [DKVKP03, DKK⁺05] introduce description

- techniques for eliciting non-functional requirements considering use cases.
- given by description techniques, specific for an application domain. For instance, Bleistein et al. [BCVP06] contribute an approach for business process and context modelling in the domain of business information systems.
- or they are developed on the basis of available description techniques gathered, e.g., from comprehensive surveys like [ZC05, NSK00, NE00].

Only few work is performed, however, for characterising projects in an isolated manner. The work of Aurum et al. [AW05a] is one example. They contribute different project parameters affecting the set up and the execution of the RE process in general, respectively the decisions to be taken in the process. Our previously performed field study [MFWLB10, MFWL⁺12] also introduces a detailed taxonomy of project parameters and their concrete effects on RE artefacts and the RE execution.

Regarding the choice of the description techniques, there exist approaches that formalise, in part, the decision problem itself. For instance, Jiang et al. [JE03] propose an algorithm for the selection of description techniques. Their work is based on their previous work contributing an N-dimensional process development methodology in combination with a multi-criteria decision making model of Zopoundis et al. [ZD00]. The latter can be compared to the tailoring matrix of the V-Modell 97 [DW99]. This matrix shows combinations of project parameters with sets of methods resulting in roadmaps that guide the process. The contribution from Jiang et al. aims, however, at the definition of a tool-supported decision support framework.

Another tendency is given by informally capturing best practices as part of domain-specific (architecture) guidelines that aim at tackling the complexity of formalised process models. For instance, the *Guide to the Business Analysis Body of Knowledge* (BABOK) [IIB09] describes a set of description techniques and the combination of these techniques according to project parameters. The outcome (different roadmaps) reflects specific industrial experiences. Similarly, the *Cooperative Requirements Engineering with Scenarios* (CREWS) project defines in a series of reports the selection and combination of scenario-based RE techniques. Rolland et al. [RP00, RPB99] then conceptualise as part of this series the guidelines by means of roadmaps, similarly as done in BABOK.

Finally, the introduced approaches focus on the syntactic layer. This means that they argue for specific methods and description techniques in dependency to project characteristics. Another area of related work also takes into account the content of requirements to be formulated in dependency to project characteristics. This area introduces approaches to reuse requirements in different projects and emphasise not specific-purpose techniques, but specific-purpose contents; for instance, approaches that informally give via patterns a collection of (textual) formulations in dependency to project goals. Franch et al. [FPQ⁺10] propose, e.g., a meta model for structuring a knowledge repository and a set pre-defined requirements formulations in dependency to project goals. Further contributions are ours in the field of security requirement, based on a quality model [LWMFB10, WMFIL09]. In fact, this field emphasises mostly non-functional requirements, since functional requirements always are subject to customer-specific, individual business processes.

Summary and Discussion. In the introduced area, related work contributes many isolated approaches, which, in part, aim at the formalisation of project parameters and the selection of corresponding description techniques, or of require-

2.6 Customisation of Development Process Models

ments contents. According to Ngo-The et al. [NTR05], however, the formalisation of decision support in isolation is unlikely suitable in practice. Many project parameters cannot be formalised, because they are of cross-cutting nature (like budget) and because strategic decisions, which often have to be taken, are unpredictable. In addition, the impacts of taken decisions on the overall development life cycle cannot be taken into account, due to the isolated nature or available approaches.

Relation to Thesis. In Sect. 5.3, we conceptualise according to Ngo-The et al. [NTR05] an approach to characterise projects and assist decision taking rather than formalising this assistance with a strict process. In contrast to available work that focusses on activity-based approaches (by characterisation and selection of methods), we transfer this conceptualisation to an artefact-based philosophy. For this purpose, we couple a set of project parameters to artefacts, which abstract from methods and techniques themselves and, thus, offer the necessary degree of flexibility and creativity during decision taking. However, because of their limited area of application, we do not take into account requirements assertions as done in approaches for requirements re-use and thereby remain to the syntactic layer.

By integrating the artefact-based reference model into the development process, we additionally enable the reflection on project parameters and their impacts on other development activities. The project parameters are gathered from our field study [MFWLB10, MFWL⁺12] and reused in the performed case studies.

2.6.4 Contributions in Development Process Models

According to Fig. 2.13 on page 50, we classify the contributions in customisation of development process models into approaches that can be allocated to the area of situational method engineering and ones which can be allocated to the area of artefact orientation. In both areas, available approaches define rules and mechanisms to customise a development process model following corresponding philosophy. We subsequently describe for both areas related work.

2.6.4.1 Situational Method Engineering

As a response to an observable tendency in software engineering to continuously develop specific-purpose methods tackling single areas of action [THV97], method engineering arises as an own research area. This area aims at the harmonisation of available methods and the systematic construction of new ones. Method engineering contributes approaches to the construction, situation-specific selection, and adaptation of methods for business information systems [Bri96].

In this section, we stick to the principles of the sub-area *Situational Method Engineering* that considers the adaptation of methods for their use in a particular project [Ode96]. The idea of situational method engineering is to retrieve, according to project parameters, appropriate methods from an administrated method repository (named *method base*). Such a repository represents a reference model that abstracts from several methods which are already customised to organisational needs and to the application domain [Ode96]. Ralyté et al. [RDR03] propose a generic process that brings together the different aspects of situational method engineering including the integration of retrieved methods (their *assembly*).

Hofstede et al. [THV97] further discuss situational method engineering with respect to its feasibility. They distinguish three major concerns:

1. *Design of Method Fragments and their Storage* for a company-wide use.

2. *Project Characterisation and Method Retrieval* in a specific project.
3. *Assembly* of retrieved methods in a project to enact a concrete project execution.

Method engineering deals with the concept of modular method fragments for the design and the storage of methods in a method base. A method fragment (in the following called method) represents a method building block that describes a systematic construction procedure to combining description techniques for creating artefacts [NE00, BWHW05]. Description techniques cover a graphical convention (notation) while the content and the structure of the artefacts are addressed only by few of available approaches [BWHW05] (see also the discussion in Sect. 3.1.1). Methods thereby describe the way of working according to a specific description technique.

The project characterisation according to project parameters (in the area named *contingency factors* [THV97]), as well as the method retrieval, aim at the selection of methods for particular projects. For this, there exist two different possibilities.

The first possibility is to define for each method different construction procedures, covering each different project parameters. Each method covers variants in its procedure to be chosen according to project parameters. This possibility is described by Fitzgerald et al. [FRO03] that present the approach used at Motorola. Such approaches are, in general, applied in the domain of standard software in which the methods to be used in the projects are all the same and already pre-defined at organisational level, but differ in their details (like differing in the used description techniques).

The second possibility is to define and store methods according to different project parameters, while no further differentiation is made. This possibility is used in the domain of custom software development projects in which the self-contained methods are selected according to project parameters. Henninger et al. [Hen98, HB01] define for this variant an approach by introducing a rule-based method repository and, thus, they build the bridge to the decision support systems, introduced the section before.

Which alternative to take affects the level of granularity in which the methods are defined [THV97]. We can define a method for its application for single techniques (e.g., “define UML activity diagram”), what is reflected in the second variant. We also can define a method for a broader area of application (e.g., “define user interaction”), what is reflected in the first variant.

Once the methods are identified and retrieved from a method base, they have to be assembled. This assembly describes the enactment of a project by combining methods in order to define a concrete process. Yoon et al. [YMB01] contribute an approach for the assembly of methods and their static verification by formalising the composed process by means of a process algebra. Most approaches, however, claim the issue of method assembly to be solved (see [THV97]), but do not take into account the syntactic compatibility of different methods w.r.t. the different incorporated notations, potentially emphasising different concepts of the application domain in a different way. Recent work, however, deals with the integration of methods, such as the contribution from Hammerschall [Ham08] and, thus, shows that this issue still is an unsolved one.

Summary and Discussion. Although situational method engineering makes valuable contributions by bringing specific-purpose approaches together, it lacks, in our context, following shortcomings:

- Method retrieval requires knowledge about strengths and weaknesses of different methods, respectively description techniques [THV97].

2.6 Customisation of Development Process Models

- The method integration (assembly) is still an issue that is not completely solved respecting syntactic compatibility and therefore syntactic consistency of the produced artefacts.
- Situational method engineering, in general, does not support the dynamic reflection on a current project characteristic respecting project parameters and the actual quality of domain-specific results with their effects of the overall development life cycle (what we consider as an objective to support a balanced problem orientation).

Relation to Thesis. We do not consider the selection and integration of methods by means of a method base. The artefact-based reference model of Chp. 4 itself serves the purpose of the method base. The static and dynamic customisation of the reference model according to the need of projects is guided by a repository that captures a set of project parameters.

This way, we tackle the shortcomings in situational method engineering. First, knowledge on different methods is not implicitly needed, since the artefact-based reference model itself makes explicit the knowledge about the concepts of an application domain. Second, the problem of variability and compatibility during the assembly of different methods is reduced by the artefact model, abstracting from methods and their relationships, and, thus, restricting methods directly in the reference model. We additionally enable the desired amount of flexibility in the choice of methods. Third, we support, on the basis of an integrated artefact model, the reflection on project parameters affecting the actual degree of completeness in the results and, thus, the impact of taken decisions on further development activities. Section 3.1 gives further information about the principles of artefact orientation and how an artefact model tackles mentioned problems.

2.6.4.2 Artefact-based Customisation

Artefact-based customisation is an area with limited contributions. We take into account one known artefact-based contribution specifically elaborated for RE and one contribution in the area of development process models. The latter describes the customisation approach used in the V-Modell XT.

Requirements Engineering Reference Model

The *Requirements Engineering Reference Model* (REM) [BPKR09, BGK⁺07, GBB⁺06] results from a research co-operation between the Technische Universität München and Siemens Corporate Research. REM defines a taxonomy-based artefact model for RE, independent of particular concepts of an application domain. The artefact model of REM is further discussed in Sect. 3.1.3, where we classify the view taken by REM on the concept of an *artefact*.

REM informally describes in addition to the artefact model a customisation approach in which the artefacts, respectively the content items of the artefacts, are classified as mandatory or optional. The customisation approach then describes

1. to choose the necessary content items that are not mandatory (“prune the artefact model”),
2. to agree on a document structure,
3. to select methods, and
4. to define the process.

REM, however, gives no concrete guidance for those steps, although it is the first known approach to bridge the gap between isolated RE contributions and customisation principles, as found in artefact-based development process models.

Relation to Thesis. Due to the domain-independent nature of REM, the customisation approach is not suitable to cover the objectives of the thesis, although many principles defined by REM serve as an inspiration for our contributions.

First, since the artefact model of REM does not take into account the basic concepts and relations of an application domain, i.e., the elements and relations found in description techniques, (1) REM must take into account methods during customisation, while (2) their assembly does not tackle the compatibility problem given in the area of situational method engineering. The contributions at hand, however, tackle this problem by reducing the variability in the choice of the methods via the definition of possible concepts and relations as part of the artefact model.

Second, REM describes customisation by means of four steps, but gives no guidance on how to perform the steps in dependency to project parameters. One difficulty is that REM has no underlying meta model, necessary to define the structural properties of the approach and, thus, the implications of one customisation step to another. Finally, the classification of artefacts to optional ones and mandatory ones is pre-defined, while their relevance arises from an application domain and from the needs of particular projects.

These circumstances barrier the re-use and adaptation of REM's customisation concepts in the context of the thesis.

V-Modell XT Customisation Approach

In Sect. 2.5.3.2, we already gave an introduction into the meta model of the V-Modell XT. The V-Modell XT [TK09] includes a stepwise customisation process, which is implemented by a tool (see also [Kuh08a]). The customisation approach [KTF10, TK09, Kuh08b, MFK09] makes use of the dynamic elements in the V-Modell XT meta model, which provide with *Project Execution Strategies* means to enact a project.

For this, the customisation approach classifies, as a first step, projects into *Project Types* that describe the frame for a particular project including a selection of chosen elements of the development process model (e.g., the project type “Customer” or “Contractor”). Project types are supplemented by *Project Type Variants* that define what *Process Modules* and *Procedure Modules* to consider and that then compute project execution strategies for the project. In addition, *Project Attributes* provide capabilities to select additional options, e.g., to embed sub-contracting parties into a project.

Process modules are the content-containing components relevant for customisation. A project-specific V-Modell XT instance is a consistent selection of process modules. Since the process modules contain all static process elements (artefacts), the artefact structure for a particular project arises from this selection. Processes and sub-processes, necessary to define a process with decision gates (milestones), are defined by the *Procedure Modules* that are selected in addition to the process modules by the project type variants. All selected modules are then used to compute the project execution strategy that, finally, states which decision gates have to be reached in which order (being enacted to a project plan).

In summary, customisation is performed in the V-Modell XT by performing a selection of relevant artefacts (from a set of possible combinations) and by defining

2.6 Customisation of Development Process Models

via the dynamic elements in which order they have to be created. This results in a project execution strategy. While the V-Modell XT offers assistance for the creation of a project execution strategy (and the definition of artefact exemplars), the creation of the artefacts' contents during the project execution is, however, not in scope.

Relation to Thesis. In contrast to RE-specific contributions, the V-Modell XT comprises a concrete and tool-supported guidance for the static customisation of the development process model according to a set of project parameters. The V-Modell XT exhibits, however, following shortcomings with respect to our objectives:

1. The customisation of the artefact model ensures consistency in-between single artefacts (e.g. documents) and to the process entities, but due to the missing notion of the artefacts' content it provides no means to support syntactic consistency in the contents.
2. The customisation approach is reduced to a fixed set of coarse-grained project parameters to be applied during initial project set-up.

Still, the customisation principles are a suitable basis, which we extend in the thesis. First, we couple, similarly as done in the V-Modell XT, detailed project parameters to artefacts, but respect also the artefacts' content and, thus, support also syntactically consistent artefacts (contents) during customisation. Second, we extend the customisation approach to the third stage (during RE execution) and give guidance to create the artefacts according to project parameters that arise during project execution. Finally, we make use of an extensible project repository to capture project parameters and their effects on the creation of the artefacts.

2.6.5 Summary and Discussion of Related Work

So far, we introduced the different stages of customisation in relation to the contributions of thesis in Sect. 2.6.1. We discussed approaches considering a process integration and approaches considering a customisation of development process models at project level. The latter was based, in turn, on a classification going beyond development process models. We discussed different approaches for customisation and took into account domain- and discipline-specific approaches, activity-based approaches with the area of situational method engineering, and artefact-based ones.

In the following, we summarise and discuss related work in direct relation to the contributions of the thesis.

Related Work for performing a Process Integration. We discussed that there is no directly related work considering the area of process integration, because a process integration eventually remains a topic to be considered by the approach that shall be integrated into a development process model. In Sect. 5.2, we define an approach to perform a process integration of our artefact-based reference model. This approach is based on our meta model. We structure the steps of our approach according to the integration approach proposed for the V-Modell XT (described in Sect. 2.6.2.1).

Related Work for performing a Customisation at Project Level. Regarding the customisation at project level, we showed that most of available work considers the initial project set-up and leaves the execution of the enacted project to

the expertise of project participants. Much effort has been spent in the area of situational method engineering complementing the area of content-centric decision support systems. These decision support systems contribute, in turn, the selection of specific-purpose methods according to project parameters.

However, due to the problem of (syntactic) compatibility of methods and the syntactic consistency of the artefacts being created during project execution, the area of method engineering has limitations in its application during the project execution.

Another problem arises from the fact that available approaches in this area

1. do not take into account the produced artefacts and their quality and, thus, do not allow for a conscious decision about whether to act during project execution in a solution-oriented way or not.
2. do not support the reflection and reaction on potentially underspecified artefacts and the impacts on further development activities that rely on those artefacts (what is especially important to RE as an integrated discipline).

In Sect. 5.3, we contribute an artefact-based customisation approach that tackles these shortcomings. In the area of artefact-based approaches, there exist two related approaches: REM and the V-Modell XT. We take the principles of customisation defined by the V-Modell XT and extend them in order to not only consider the set-up of the process, but also its execution.

For this, we make use of a multi-project environment for characterising projects according to project parameters. In contrast to available approaches in the area of decision support systems, which characterise projects by the impact of project parameters on different methods and description techniques, we transfer this idea to the artefact-based philosophy. The parameters guide, similar to approach of the V-Modell XT, the enactment by creating exemplars of the artefacts and putting those in an ordered sequence for the delivery with respect to a set of milestones. During the RE execution, however, we also support the reflection on fine-grained project parameters and the appropriate content creation within the enacted process frame with respect to an appropriate degree of completeness in the artefacts. In contrast to available approaches, we additionally enable (due to the process integration) the reflection on the impacts the created artefacts have on other activities of the development life cycle and, thus, support a balanced problem orientation.

2.7 Introduction of a Running Example

In this section, we introduce a running example to which we refer in the different contributions of the thesis.

The example considers a software development project within the fictitious company *Alpine Adventure Tours* (AAT). The company targets the tourism market in the winter sport region in southern Germany and has an information systems to manage different skiing courses. Since the company aims at a market expansion and collaboration with further companies, the historically grown course management systems does no longer support the new business processes. As a consequence, the software development project aims at

1. a re-design of the business processes to meet the new business needs, and
2. a replacement of the course management system for which the corresponding requirements shall be specified.

2.7 Introduction of a Running Example

CHAPTER 3

Artefact Orientation

Artefact orientation has many facets whereby there still is no common understanding and agreement on structure and semantics of an artefact-based approach and, in particular, on the term *artefact*. There are interpretations in the granularity range from general structures of documents or data sets to detailed models, which abstract from modelling concepts for a particular application domain. In fact, the design of an artefact model depends on the intended purpose of the model affecting consequently the design of an artefact-based approach. In the following chapter, we thus analyse, as a first step, the principles of artefact orientation and differentiate those principles from the ones followed by activity-based approaches. We discuss different views that can be taken onto an artefact model according to general objectives potentially followed with artefact orientation. Based on our particular research objectives that aim at an artefact-based customisation, we then infer the necessary design of our meta model artefact orientation. This resulting meta model is then presented in the third section. We conclude with a discussion on how we ensure the validity during instantiation of the meta model to the application domain of business information systems. The content of the chapter at hand is based, in part, on our contribution in [MFPKB10].

At the end of this chapter, the reader will understand (1) how an artefact-based approach can be designed according to different objectives, (2) how it should be designed to achieve our research objectives, and finally (3) how our objectives result in a meta model for artefact orientation. This meta model lays the foundation for the construction of the domain-specific approach in Chp. 4 and for the definition of the customisation approach in Chp. 5.

Contents

3.1	Principles of Artefact Orientation	60
3.2	Design of a Meta Model for Artefact Orientation	73
3.3	Meta Model for Artefact Orientation	77

3.1 Principles of Artefact Orientation

The idea of focussing on artefacts in development processes is not new. In the model-based design and development philosophy, an artefact is seen as an abstraction from modelling elements used as an intermediate result of a process. An artefact has particular properties and may be described using standardised modelling concepts. One is able to incorporate different description techniques and notions for producing the artefacts and build on a customisable process. Project participants agree on the artefacts to be delivered and do not have to care about the realisation process in volatile project environments. The diversity in the processes is reduced to the dependencies between the artefacts themselves without having to take into account the complexity of differing processes. Thus, the complexity of a development process is reduced on the basis of a pre-defined result structure (see also introductory Sect. 1.3.2).

Hence, artefact orientation has also become a well-accepted technique in the area of development process models. Available models give evidence on the evolution from activity orientation (e.g., reflected in the *V-Modell* 97 [DW99] or the *Rational Unified Process* [KK03, JBR99]) to artefact orientation incorporating the process-agnostic philosophy; *Prince 2* [oGCO09], as a reference model for IT management, or the *V-Modell XT* [FHKS08] (see Sect. 2.5.3.2) are prominent examples.

Although artefact orientation has become a well-accepted technique, there is, however, still no common understanding and agreement on syntax and semantics of artefact-based approaches. The major reason is that the view taken on an artefact model, being the backbone of a development process model, respectively the view taken on the structure of an *artefact* itself, depends on the intended purpose of the artefact model.

In order to get a unified understanding of artefact orientation necessary to achieve our research objectives, we define a meta model for artefact orientation. To define this meta model, we analyse, as a first step, the major principles of artefact orientation and different views that can be taken. We conclude with a discussion of the variations of artefact models and their different characteristics. Based on this discussion, we infer in the following sections 3.2 and 3.3 a meta model for artefact orientation with respect to our particular research objectives.

This meta model shall support process integration and customisation of the domain-specific instance of the meta model, while the instance itself (for the application domain of business information systems) is defined in Chp. 4.

3.1.1 Artefact Orientation versus Activity Orientation

In Sect. 2.5.3, we already gave a first impression of the difference between artefact orientation and activity orientation. In the section at hand, we discuss the basic principles of both philosophies by directly comparing them.

For this, we take into account method engineering (see Sect. 2.6.4.1) in addition to activity-based development process models. Although there exist concrete development process (meta) models that are defined following, e.g., the activity-based philosophy (see Sect. 2.5.3.1), we discuss in this section the basic principles of the philosophies. We thus exclude any discussions and evaluations of available development process (meta) models. An introduction into advantages and disadvantages of activity-based and artefact-based approaches has already been given in the introductory related work section 1.3.

Figure 3.1 depicts a simplified view taken by both philosophies on the different sub-models of a development process model. We depict the the concepts *Role*, *Arte-*

fact, *Activity* (with methods), and *Description Technique*, which constitutes means for the representation of an artefact in a particular shape (see also Sect. 2.5.1.2 discussing the sub-models of a development process model).

The upper part of the figure illustrates the arrangement of the sub-models of a development process model following the activity-based philosophy; the bottom part of the figure illustrates the arrangement following the artefact-based philosophy.

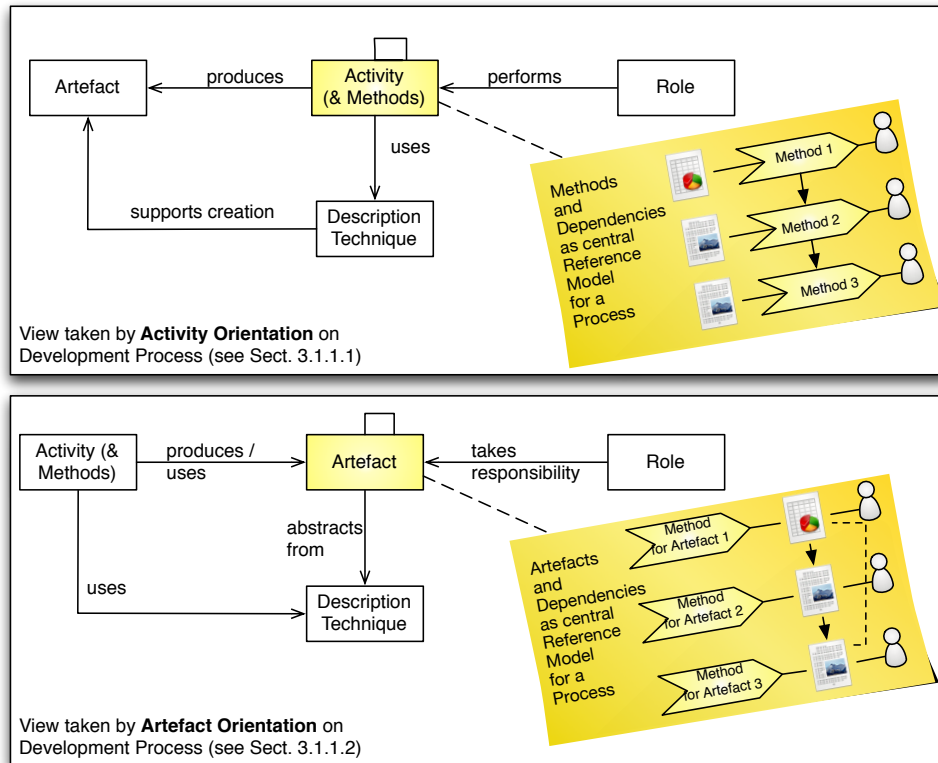


Figure 3.1: Comparison of philosophies by their view taken on the sub-models of a development process model and their arrangement (simplified).

In the following, we describe both philosophies according to this figure and give afterwards a concluding summary.

3.1.1.1 View taken by Activity Orientation on Development Process

Braun et al. [BWHW05] analyse available approaches following the activity-based philosophy. They discuss and compare the conceptualisation of the sub-models of a development process model found in different method construction approaches, including representative approaches of Brinkkemper [Bri96], Gutzwiller [Gut94], and Balzert [Bal98]. On the basis of this comparison and taking into account available development process meta models that follow the same philosophy, we organise the sub-models of the development process model in the upper part of Fig. 3.1. Accordingly, activities (respectively methods) are seen as “construction tasks, which create certain results” [BWHW05], being performed, in turn, by a role. Activities build in this perspective, in general, the backbone of a development process model, because a concrete project execution arises from a specific sequence of activities. Each of those activities then makes use of several description techniques in order to produce the desired artefacts. The artefacts are previously de-

3.1 Principles of Artefact Orientation

defined and structured specification documents of a specific type in which the results are recorded in dependency to the chosen description technique [BWHW05]. While activities have in available approaches a well-defined structure to provide an approach to combining different description techniques [NE00], only 50% of the approaches, analysed in [BWHW05], include the definition of an artefact sub-model at all.

Hence, the view taken by activity-based approaches emphasises how to do something rather than defining what has to be produced. As a consequence, domain-specific approaches that are built upon such principles, like the ones for enterprise architecture modelling [FBB09] or for modelling of service-oriented architectures [Off08], offer a vague description of the content and structure of the artefacts to be produced. The same consequence can be observed in comprehensive development process models, like the ones that are based on SPEM (see Sect. 2.5.3.1).

Although the importance of a well-defined artefact model is recognised in the area of activity orientation [FBB09], artefacts are still reduced to an outcome of activities and underlie an optional description in available approaches. The emphasis lies on the definition, selection, and integration of methods, ready to be applied at project level.

3.1.1.2 View taken by Artefact Orientation on Development Process

Compared to activity orientation, where the process is built on basis of a model of different activities and their dependencies, artefact orientation follows a process agnostic philosophy. The emphasis of artefact-based approaches lies on consistent result structures and used terminology, defined by means of an artefact model. The other sub-models of a development process model are coupled to the artefact model. One gives attention to the deliverables (and their dependencies) to be produced by particular roles, and exchanged as an intermediate of different activities and, thus, one gives attention on what has to be produced rather than on how it has to be produced.

Artefact models, however, are mainly subject to two major areas of application: to the area of development process models and to the area of model-based design and development. Although the latter area disregards the definition of a development process by means of activities and roles, we take into account both areas for the arrangement of the different sub-models in Fig. 3.1.

Regarding development process models, with a prominent artefact-based approach given by the V-Modell XT (see Sect. 2.5.3.2), all relevant sub-models are coupled to the artefact model being the main source for defining a concrete project execution. Accordingly, an artefact model abstracts from concrete realisation details (activities), since all parties agree in a first step on the artefacts to be produced and then build a process upon these artefacts. Roles are assigned and allocated to an artefact for which they take the responsibility, and activities are selected after agreeing on the relevant set of artefacts. The concrete project execution automatically arises from the customised artefacts and their dependencies, being defined as part of the artefact model.

In addition to the area of development process models, we take into account the area of model-based development in which an artefact model abstracts from description techniques (see for example [SPHP02]). In this area, an artefact model is used as a domain-specific reference model giving guidance for the use of different modelling languages. More precisely, the artefact model guides in a project the creation of concrete models for generating, in turn, source code. As a matter of fact, the area of development process models sees an artefact as anything that

can be an outcome of activities, including for example the system to be delivered. The area of model-based development sees an artefact as an exclusive abstraction of description techniques and a system under consideration (see also Sect. 2.5.2 discussing possible views that can be taken at different levels of abstraction). We take, however, into account that each artefact underlies a description, even if in a generic manner, whereby the view given in Fig. 3.1 satisfies the depiction of the basic concepts of artefact orientation.

3.1.1.3 Comparison of Activity Orientation and Artefact Orientation

Table 3.1 summarises the characteristics of the activity-based philosophy and the artefact-based one w.r.t. the views given on the basic concepts and their arrangement.

Table 3.1: Activity-based versus artefact-based philosophy.

	Activity Orientation	Artefact Orientation
View on Activities	<ul style="list-style-type: none"> • Profound definition of activities, methods, and dependencies • Process via arrangement of activities 	<ul style="list-style-type: none"> • Model-based dev. approaches neglect process at all • Development process models: <ul style="list-style-type: none"> – Methods optionally (flat method structure) – Process via arrangement of artefacts to be created
View on Artefacts	<ul style="list-style-type: none"> • 50% of approaches neglect artefacts • Artefacts part of self-contained methods • Flat artefact structure with weak content information and no relationships 	<ul style="list-style-type: none"> • Artefact model comprehensively defines artefacts and dependencies • Centre of development process models (roles & activities coupled to artefacts) • Structure and content (level of detail) of artefacts differs in dependency to area of action

Activity-based approaches put emphasis on activities and build a concrete process by combining selected activities. Artefacts are in most of available approaches defined as an optional element. In approaches, which define an artefact as mandatory, however, they are reduced to an exclusive outcome of self-contained methods that produce an artefact (see also a representative meta model given by Hammer-schall [Ham08]). Hence, the relationships between the artefacts are not defined, respectively reduced to relationships between the activities. While activity-based approaches have the advantage of offering concrete instructions for defining new and applying existing methods in a project, the processes become complex and the approaches do not effectively cope with the various influences given in individual project environments. In addition, we have to face the problem of syntactic inconsistencies, since the dynamic combination of methods implies the problem of a potential syntactic incompatibility of different description techniques.

Artefact-based approaches, in turn, reduce the complexity in the process by restricting the possible structure (and content) of the results, whereas the process is defined by deciding on what artefacts to produce. Since the sub-models of a development process model are all coupled to an artefact model, the decision about what artefacts

3.1 Principles of Artefact Orientation

to produce automatically gives guidance on what activities to perform. Since one can choose for any method that offers means to create the elements and relationships defined in the artefact model, corresponding approaches reduce the possibilities of creating the results in an incomplete manner. Still, although artefact-based approaches support a variable, integrated, and consistent view onto development processes, available approaches have to face the contrary problem of activity-based approaches. Artefact-based approaches exhibit the characteristic that (1) methods are often underspecified or do not exist at all and (2) the result structure is fixed and hardly extensible, since artefacts always abstract from already existing methods or description techniques.

While there exists a common agreement on structure and content of activity-based approaches, there still is no common agreement on the one of artefact-based ones. The reason is that the understanding of artefact orientation arises from the different views that can be taken in dependency to the different areas in which artefact models are used; for instance, the view taken by the area of model-based development or the one taken by artefact-based development process models.

In fact, how to define an artefact model and how exactly to set this artefact model in relation to the other sub-models of a development process model depends on the view taken. This view arises, in turn, from the objectives followed during the establishment of the artefact model. In the following, we discuss a selected set of possible objectives.

3.1.2 Potential Objectives in Artefact Orientation

Artefact models are used in different areas, ranging from the field of model-based development over general development process models to the use as a domain-specific ontology. In this section, we discuss selected objectives that are potentially followed when establishing an artefact-based approach. Based on these objectives, we discuss in the following sections resulting views on an artefact model, i.e., the way the artefact models are defined and used.

We distinguish the following objectives:

1. Support of project management
2. Flexibility for customisation
3. Explicit representation of domain knowledge
4. Consistency and completeness of created results
5. Seamless modelling
6. Tool-support (operationability)

Support for Project Management. One aspect considered by project management is the definition of responsibilities in a project. This objective is achieved in artefact orientation by directly coupling roles, respectively specific team members, to artefacts for which they have to take the responsibility (see Fig. 3.1). Another aspect important to project management is the establishment of progress control. Progress control is supported, because quality assurance metrics can be defined for objectively measuring the degree of completeness of an artefact w.r.t. the artefact-based reference model, which defines a notion of structure and content in the artefacts. Based on assessable completion levels, the actual status (e.g., “finished”) of an artefact can be determined to support the overall progress control.

Flexibility for Customisation. The flexibility for customising a process is an important topic in the area of development process models. We consider a reference

model as flexible if it allows, at organisational level, for a systematic process integration, and if it allows, during its application at project level, for variations in the created artefacts in response to project characteristics. Process integration and customisation at project level are both supported by the artefact-based philosophy, since artefact models are used to reduce the variability in the processes (see the foregoing section).

Explicit Representation of Domain Knowledge. The use of artefact-based reference models supports the establishment of a clear terminology for a particular application domain. Since artefact models capture the basic (modelling) concepts of the domain, they make knowledge about the concepts of the domain explicit. Thus, they tackle the shortcomings of activity-based approaches, which, in contrast, presume knowledge about weaknesses and strengths of different methods (see Sect. 2.6.4.1 discussing the limitations of method engineering).

Consistency and Completeness of created Results. In the area of model-based development, artefact models are used to capture the modelling concepts and their relations of an application domain as a model blueprint so that they serve as orientation for creating project-specific results in a syntactically complete and consistent manner. If the artefacts created at project level allow for simulations, they additionally can be used to identify semantically inconsistent or incomplete results.

Seamless Modelling. A prerequisite for seamless modelling is that the different models, created at project level, are consistent, i.e., that the elements and relations to be created are known in advance and preserved during their creation, e.g., in a tool-supported manner. Since artefact models can be used as such reference models, they thus support seamless modelling and consequently continuity in the development process.

Operationability via Tool Support. Artefact models can be defined as reference models to be manually used, for example, to create conformant results in a project, but also to apply tool support in project environments. Regarding the latter, artefact models, which then are defined as data models made available for computation, aim at the definition of elements and relations to be considered during (tool-supported) modelling activities. The artefact models then achieve an operationalised domain knowledge in order to ensure the conformance of results being created in projects to the structure and the content of the artefact-based reference model. For instance, by applying consistency rules to be used for tool-supported validation and verification.

3.1.3 Potential Variations of Artefact Models

According to Davis, “the value of a model depends on the view taken, but none is the best for all purposes” [ER03]. In case of artefact models, there exist different interpretations of the corresponding concepts, each following particular objectives. In the foregoing section, we introduced the different objectives (respectively purposes) that can be followed during the establishment of an artefact model. In this section, we analyse the different views that are taken on the notion of an artefact model in direct response to the different objectives.

3.1 Principles of Artefact Orientation

Although there exist approaches that mix different views as part of one model, we identify the following variations of an artefact model into which we group available approaches (see Fig. 3.2):

1. *Structure models*, which exclusively define the general structure of specification documents or data sets, e.g., as a taxonomy.
2. *Content models*, which describe the content of specification documents or data sets, e.g., by defining with a data model the modelling concepts used to create the contents of specification documents. The content models can be either defined in a generic manner or for a particular application domain.
3. *Integrated modelling theories*, which capture the underlying theory of domain-specific modelling concepts, e.g., with precise mathematical models.

View	Structure Model	Content Model					Integrated Modelling Theory
		Generic Content Model		Domain-specific Content Model			
Exemplary Representation	Taxonomies	Checklists	Guidelines	Ontology Architecture Models	Concept Models	Concept Models enriched with Conformance Constraints	Mathematical Models

Figure 3.2: Variations of artefact models.

In the following, we characterise, according to Fig. 3.2, the different views and exemplary means for representing these views, as depicted by the second row of the figure. For each representation, we discuss available approaches. In Sect. 3.1.4, we discuss the different artefact representations in relation to their characteristics and how they achieve the objectives stated in Sect. 3.1.2.

3.1.3.1 Structure Models

Structure models define a reference model of topics to be considered in documents or data sets. A structure model is mostly represented as a *taxonomy*-based reference model, which defines the hierarchical structure of, for example, a specification document and its chapters; for instance, by defining an artefact *Requirements Specification* including topics like *Use Case Model* or *Functional Requirements* that each can be decomposed into further topics. Structure models, in general, emphasise the structure of the results and aim at a common, standardised understanding of what should be generally produced within individual projects. The area in which structure models are mostly applied is given by development process models. Since this coarse-grained view does not include a detailed description of the contents (e.g. the content of a use case model and its dependencies to functional requirements), the artefact models offer a low degree of complexity with respect to the dependencies between the artefacts and, thus, they facilitate their process integration and customisation (“flexibility”).

Exemplary Approaches. One exemplary approach that makes use of a structure model for the definition of an artefact model is the development process model *V-Modell XT* [FHK08, VMX]. The V-Modell XT defines all artefacts (respectively product types) to be used or produced as part of a development project. Due to

the abstract view on artefacts including dependencies among the artefacts, the process integration and the project-specific customisation are both supported. Further information on the V-Modell XT can be taken from related work in Sect. 2.5.3.2. However, the V-Modell XT defines a flat artefact structure with all deliverables of a process and makes weak assertions on the topics to be considered within each artefact (e.g., within the requirements specification document), since the V-Modell is intended to be generic and thereby applicable for different domains of application. A more detailed structure model, but with a limited scope, is given with the *IEEE Std. 830-1998 recommended practice for software requirements specifications* [IEE98]. This standard gives a reference structure model for requirements specification documents and describes different topics to be considered (independently of an application domain), as well as different possibilities of arranging these topics when using this reference model in a project.

Discussion. The introduced approaches describe artefact models from an abstract view. Although they include the general description of topics to be considered as part of a taxonomy, they do not consider the content of the artefacts in terms of, e.g., content descriptions of the topics and what description techniques can be used. On the one hand, this supports flexibility for customisation. On the other hand, there is no guidance for creating the contents. This guidance is given, in turn, by content models. We subsequently describe the view taken by content models; first, in a generic manner and then for a particular application domain.

3.1.3.2 Generic Content Models

Generic content models define the content of artefacts, independently of the chosen artefact structure when recording the content. Such models give guidance on the elaboration of the artefacts; for instance, by describing what a use case model in general is and by (optionally) referring to possible description techniques. We distinguish for the representation of generic content models between *checklists* and *guidelines*. While the first is a taxonomy-based model that describes for each topic best practices that should be considered, guidelines structure the content (semi-formally) from a methodological point of view. Guidelines arrange the content by following generic refinement-principles, e.g., by describing how to first elaborate goal models and how to then derive a use case model.

Exemplary Approaches. An exemplary approach for a checklist, specifically elaborated for requirements engineering, are the *Volere requirements specification templates* [RR07]. Examples that capture the content from a methodological point of view in guidelines are given by contributions made by Wiegers [Wie03], or by the *Requirements Engineering Reference Model (REM)* [BPKR09, BGK⁺07, GBB⁺06] (see also Sect. 2.6.4.2 where we give a first introduction into REM and its customisation approach). REM defines, independently of an application domain, the contents to be considered, including goals, requirements, and system specifications. This is done within a proposed taxonomy-based guideline that informally describes dependencies between the elements of the guideline according to proposed refinement principles.

Discussion. The introduced approaches use checklists and guidelines to establish a generic reference of the general content and relations to be considered. Still, those models provide no concrete information about the possible concepts like

3.1 Principles of Artefact Orientation

the ones for constructing use case models and, thus, they do not formally capture the interdependencies between the concepts; for instance, the dependency of use case models to the ones of functional requirements. Although these models provide contributions to support a common understanding of the basic principles of disciplines like RE, they do not provide a reference model serving as orientation to make use of domain-specific concepts in a syntactically complete and consistent manner. This leads to the next view given by domain-specific content models (see the next section). However, note that REM is the first known RE approach that incorporates the artefact-based philosophy considering also roles and methods. REM was taken as an inspiration for two domain-specific contributions: REMsES [PSP09] for the domain of embedded systems¹ and Quasar Requirements [MFC09]. The latter represents our (industrially hosted) previous work in the development of an artefact-based approach for business information systems and is extended in the next chapter according to our particular objectives. Section 3.2 gives further information on our previous work and the relation to the thesis.

3.1.3.3 Domain-specific Content Models

Domain-specific content models capture the basic concepts of a particular application domain, such as the one of business information systems. These models tackle the problems of generic content models by describing the necessary content of documents and data sets respecting methodological aspects, i.e., refinement principles, for a concrete application domain. Hence, these content models give concrete guidance for the elaboration of the content and enable awareness of domain-specific concepts, relations, and terms.

As depicted in Fig. 3.2, we distinguish between three different representations for the establishment of domain-specific content models.

The first representation is given by an ontology as part of an architecture model. An *ontology architecture model* captures the basic concepts and their relations for a particular domain with a particular focus on architecture principles and related terms; for instance, for service-oriented architectures (see also Sect. 2.3.2). A further representation is given by concept models.

Compared to an ontology architecture model, a *concept model* raises the degree of precision in the content description and describes, mostly as part of a data model, elements and relations given in different description techniques. Concept models abstract therefore from domain-specific methods and description techniques that “define those aspects of a system under consideration dealt with during the development process” [SPHP02]; for example, by defining of what elements a use case model exactly consists and what relations (and cardinalities) should be captured to further concepts like the one of a functional requirement. The defined modelling concepts then include concrete types and dependencies and offer, besides a formal syntax, bits of semantics to define the artefacts content and to give guidance for their creation. The use of concept models as a reference model therefore supports awareness of creating domain-specific results in terms of supporting syntactic completeness and consistency during modelling activities. By representing concept models as data models they also provide the possibility to explicitly specify the desired amount of redundancy of contents. Thus, one is also able to establish comprehensive tool support on this level, because the data models are available for computing.

¹ See also <http://remses.org>

Schätz describes in [Sch08a] the possibilities of additionally enriching domain-specific *concept models with conformance constraints*. Thereby, he supports the quality of, for example, requirements specifications in terms of ensuring syntactic conformance of the models being produced within a project to the reference model in a tool-supported manner. In fact, the application of conformance constraints is mostly used in the area of tool-supported quality assurance. How conformance constraints are defined, therefore also depends on the chosen quality assurance technique (and the used tools).

Exemplary Approaches. Approaches that make use of an ontology architecture model are, e.g., the *Zachmann framework* [Zac87] or industrial derivatives like the *Capgemini IAF* [Gro06a] (see also Sect. 2.3.2 introducing architecture frameworks for business information systems). An exemplary approach that emphasises methodological aspects, e.g., for service-oriented architectures, is the *SOA Ontology* of The Open Group [Gro09b].

Since approaches that include concept models can be allocated to the area of model-based development, covering, in parts, the development and integration of tools, there exists a variety of available approaches. These approaches are defined for different domains of application and have a limited scope due to their envisioned area of application. An example for a concept model is given by the data model of *AutoRAID* [SFGP05], a modelling tool that is situated in the domain of embedded reactive systems. Another example for a concept model is given by the contribution from Berenbach et al. [BW07] that couples concepts of use case modelling and feature modelling, or by the contribution from Zhang et al. [ZYCJ06] that defines concepts for modelling services, both describing blueprints of applied single techniques considering an integrated tool chain. A more comprehensive view is taken by approaches that emphasise the concepts used in design activities of whole application domains, such as for the one of web systems being, e.g., described by the approach *UWE* (UML-based Web Engineering) [KKZB08]. The last contribution to be mentioned is from the Object Management Group (OMG) with the *SoaML* [Ber08] for service-oriented architectures and UML-based specifications in general.

Finally, an example for the definition of conformance constraints is given by Schätz [Sch01] using a logic-based formalism.

Discussion. The introduced approaches offer domain-specific reference models that define the basic concepts, relations, and terms of a particular application domain. Concept models, which abstract from domain-specific description techniques, support awareness of modelling the contents of, e.g., specification documents in a syntactically complete and consistent manner. Still, available approaches have shortcomings.

First, the approaches cover a limited scope such as the content of a particular specification document (mostly of design documents). In addition, these approaches make mostly use of UML-based modelling languages using UML profiles and focusing on the model-driven methodology in system design (solution design). They do not take into account other development phases like RE and the dependencies between different phases. Second, many concept models abstract from available description techniques that offer specific-purpose approaches for a particular area of action while the notions have limited underlying semantics.

This lack of semantics allows for ambiguous interpretations of the concepts what, in turn, leads to the most formalised view on an artefact model given by integrated modelling theories.

3.1 Principles of Artefact Orientation

3.1.3.4 Integrated Modelling Theory

A problem with concept models is that they often abstract from available description techniques that

1. are defined as specific-purpose solutions to limited areas of action,
2. do not consider all facets of systems and development activities due to their specific purpose, and
3. give no concrete guidance for their application and interpretation.

Especially the missing guidance for the interpretation of applied description techniques arises from those techniques that put emphasis on their pragmatic application rather than on the underlying semantic foundation of the used concepts and relations (see also the discussion in the section before).

In addition to the missing semantic foundation of the used concepts, the relations between those concepts are defined from a pure methodological point of view. The UML is, in parts, one prominent example that gives evidence of this problem. On the one hand, the different diagram types have been discussed for a long time to be the lingua franca in modelling because of their wide area of application and their pragmatic use. On the other hand, much effort has been spent for formalising single description techniques (particularly for modelling behaviour), since the semantic foundation of these techniques is not given.

An integrated modelling theory tackles, however, this general problem while giving the most formalised view on artefacts (see also Broy et al. in [BFH⁺10]). In such a theory, formalisation of concepts and relations found in the description techniques is performed by means of, for example, precise *mathematical models*. Such a model aims at the definition of a semantic foundation of the techniques, e.g., by defining properties of modelling concepts like “use cases” and (mathematically expressed) relations to other concepts like “services”. A modelling theory thus provides a set of modelling concepts and means to capture various aspects of composition and architecture [BFH⁺10].

In contrast to concept models, one thereby puts during the development of a modelling theory emphasis on the semantics of concepts that capture all facets of systems and development activities in a particular domain rather than on potential description techniques and their pragmatic application potentially covering only selected (system) concepts².

Finally, since the used mathematical models offer their own syntax or are used as a basis for the definition of the like with a precise mathematical meaning, syntactic consistency is enforced.

Exemplary Approaches. One example for an integrated modelling theory is given by the *Focus* theory [BS01]. Focus offers concepts for defining and structuring functionality by function hierarchies, describing dependency relations, and techniques to model functions with their behaviour (including time behaviour) in isolation. Concepts for composition and refinement are means to describe an architecture including syntactic interfaces and behaviour interfaces [BFH⁺10].

Based on this modelling theory, several extensions are available, such as extensions for modelling business processes [Thu04] or services [BKM07], both emphasising functional aspects and offering an own syntax.

Modelling theories are, in general, available to

- precisely construct the basic concepts, relations, and terms for a particular application domain in a descriptive manner, as done in our contribution for SOA [BLMF⁺10], and to

² In fact, we can establish a concept model based on a theory rather than vice-versa (see also Sect. 3.3.3).

- construct a semantically sound seamless engineering approach including an integrated tool chain, as for example done by Leuxner et al. [LSS10].

Discussion. Theories tackle the problem of modelling languages with a weak semantic foundation. By defining concepts and relations via mathematical models and by offering an own syntax, syntactic completeness and consistency is enforced. However, although theories provide a basis for seamless modelling and an integrated tool chain, the applicability is by now hampered in two ways. First, available theories emphasise the formalisation of functional (behavioural) properties of systems and modelling concepts and non-functional properties that are directly related to behaviour (such as time constraints). Second, the pragmatic use and interpretation of theories, respectively mathematical models, demands for expertise and domain knowledge and / or depends on available tools (see also [BLMF⁺10] for further discussions).

3.1.4 Variations of Artefact Models according to Objectives

Figure 3.3 illustrates the possible variations of an artefact model introduced in the section before: Structure models that exclusively define the structure of documents or data sets; content models that define the content of artefacts either in a generic manner or for a particular application domain; and modelling theories that define the underlying semantics of concepts and terms used in the content models.

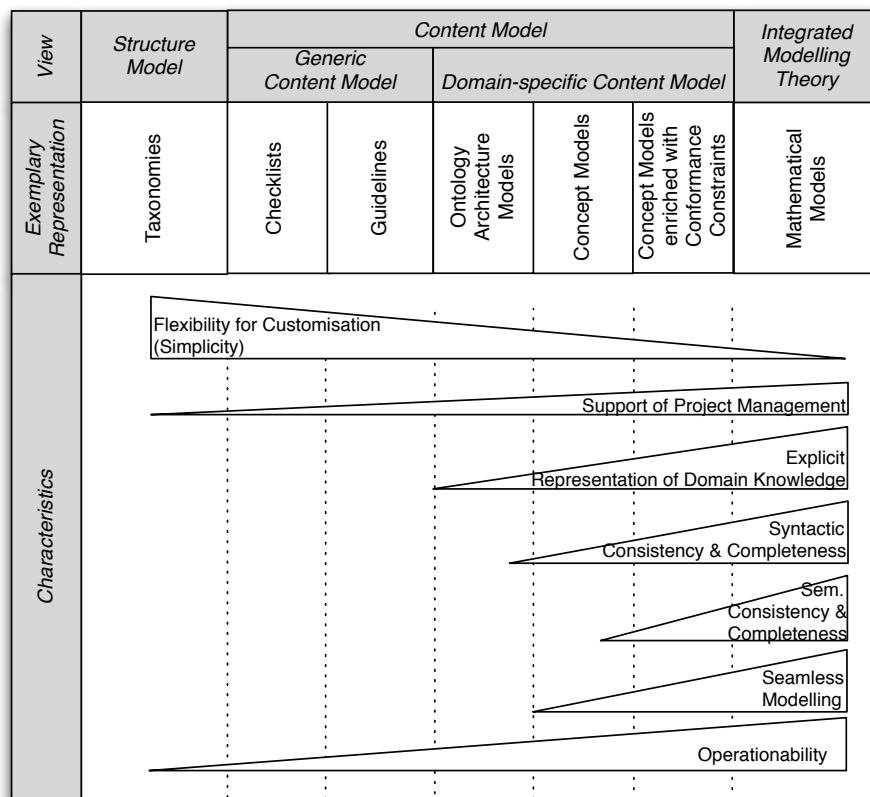


Figure 3.3: Variations of artefact models with characteristics.

3.1 Principles of Artefact Orientation

Content models imply always a notion of structure, as seen in the contribution from Schätz [Sch08a] that provides means to hierarchically (de-)compose document structures as part of the concept model itself. In addition, concept models can be built on the basis of modelling theories (see also Broy et al. [BFH⁺10] and Sect. 3.3.3.2). Still, different objectives justify the use of a self-contained representation of an artefact model.

For this reason, we now extend the introduced variations of an artefact model with their corresponding characteristics (depicted in the third row) and discuss how the different variations achieve the possible objectives stated in Sect. 3.1.2.

In general, approaches that demand for a high applicability and *flexibility for customisation* are built on abstract (hierarchical) structure models exclusively represented as taxonomies. The reason is that structure models abstract from realisation details and represent whole topics that can easily be process-integrated; for instance, by establishing an association from a single element “use case” to corresponding methods for producing the artefact and to methods that make use of the artefact. Moreover, structure models can be customised at project level according to varying project parameters that affect the creation of single artefacts or included topics. Instead, in order to *support project management* w.r.t. progress control, artefacts need, in part, a notion of content so that the actual degree of completeness of an artefact can be objectively measured.

Going in Fig. 3.3 from left to right, we see in the different representations of an artefact model a decreasing degree of simplicity w.r.t. the artefacts’ contents. The more precise we define the content of artefacts, the lower the flexibility when using these artefacts in a reference model. Instead of defining an abstract element “use case” as part of a taxonomy, we would define each single element found in corresponding description technique for producing a use case; for instance, an action, a concept for actors, processed information objects, and the relations between those elements.

Although having to administrate the higher complexity, a detailed domain-specific content model benefits the *explicit representation of domain-knowledge*. Especially the concept models and the mathematical models support the awareness of domain-specific concepts, relations, and used terms.

Both the concept models and the mathematical models, abstracting from (and restricting) description techniques, benefit syntactic consistency and completeness of the results being produced in conformance to the artefact model. Hence, such models enable *seamless modelling* and, thus, continuity in the development life cycle because the process elements that are coupled to the artefact model provide a continuous work flow.

The syntactic characteristics and seamless modelling are, however, enforced when using mathematical models rather than concept models. The reason is that concept models are not per se complete, because they always abstract from existing description techniques with possibly incomplete concepts (emphasising the pragmatic use rather than precision on the syntactic layer). This barrier can be reduced if building a concept model on the basis of semantically founded description techniques.

Semantic consistency and completeness can never be guaranteed [IEE98], but at least supported up to a certain level. Concept models and mathematical models can be made available for computation. This allows, although restricted to functional behaviour models and non-functional aspects that are directly related to such models (like time-behaviour), the simulation of models. Such a simulation benefits the validation, e.g., by detecting semantically incomplete behaviour models and contradictory assertions.

Finally, the *operationability* is achieved by each view. The tool-supported assurance of the syntactic conformance of the artefacts can be achieved with concept models that are enriched with conformance constraints (like consistency rules), and with mathematical models.

3.2 Design of a Meta Model for Artefact Orientation

There exist two variants for the design of a meta model for artefact orientation. First, the meta model can be designed from scratch in a constructive research approach. This variant is usually performed when exploring new domains according to particular requirements. Second, the meta model can be inferred (reverse engineered) by analysing the elements of given domain-specific reference models. In this thesis, we follow a combination of both variants.

As already mentioned in Sect. 3.1.3.2, we developed, inspired by approaches like REM, first parts of our contribution of Chp. 4 in an industrial environment. In Sect. 3.3.3, we give further information on the development of this initial approach to demonstrate how we ensure the validity of our contribution. Based on this domain-specific reference model and another model for the application domain of embedded systems (REMsES) [PSP09], we inferred an initial meta model for artefact orientation in [MFPKB10]. This meta model should give a unified guidance for future work on artefact-based RE in the research group of Software and Systems Engineering at the Technische Universität München.

We now take, however, these previous contributions as a set of lessons learned into account and systematically construct step by step the meta model according to the objectives of the thesis.

So far, we discussed the principles of artefact orientation and different views that can be taken on an *artefact model* depending on different possible objectives. In this section, we discuss our particular research objectives and infer a set of requirements for the design of our meta model. Based on those requirements, we describe the necessary view on an artefact model and its interrelations with further sub-models of an artefact-based development process model. In Sect. 3.3, we then introduce the corresponding meta model for artefact orientation to be used in Chp. 4 for the domain-specific instantiation.

3.2.1 Research Objectives and Requirements for Artefact Orientation

In the introductory Sect. 1.2, we defined the research objectives of the thesis. These objectives consists in defining a systematic approach that

1. includes an RE reference model supporting awareness of syntactically consistent and complete results with respect to the basic concepts of an application domain, and that
2. ensures flexibility for a process integration and for the customisation of the RE reference model at project level.

In the following, we describe the necessary characteristics of our artefact-based approach to infer a set of requirements on the meta model for artefact orientation. According to those requirements, we construct afterwards in Sect. 3.2.2 the meta model for artefact orientation.

3.2 Design of a Meta Model for Artefact Orientation

3.2.1.1 Necessary Characteristics of an Artefact-based Approach

In order to satisfy the objectives stated above, the artefact-based approach has to achieve the following characteristics of the ones we introduced in foregoing Sect. 3.1.4.

The artefact-based approach has to support the *explicit representation of domain knowledge* in order to support awareness of the basic concepts, relations, and used terms in the artefacts created for the envisioned application domain. Thus, the concepts and relations used in the reference model have to include an explicit notion of structure for the representation of the content of documents and data sets.

When applying this reference model at project level, it has to guide the creation of the artefacts in a *syntactically consistent and complete* manner.

To support the application of the reference model, we need to ensure the *flexibility for customisation*, i.e., we have to allow for variations in the created artefacts in response to individual project characteristics. At organisational level, we need to ensure flexibility in terms of allowing for a systematic process integration of the reference model into a development process model.

Moreover, for a systematic RE approach as an interconnected discipline, the approach has to *support project management*. In particular, we need to support the definition of responsibilities for the creation of artefacts and progress control during this creation.

3.2.1.2 Resulting Requirements for the Meta Model for Artefact Orientation

In the following, we formulate 5 requirements for the meta model for artefact orientation in order to achieve the characteristics and research objectives stated above.

REQ 1. *Flexibility for Customisation by defining a Structure Model*

As discussed in Sect. 3.1.2, we consider a reference model as flexible if it allows, at organisational level, for a systematic process integration and if it allows during its application at project level for variations in the created artefacts in response to project characteristics. As shown in Sect. 3.1.4, we can support this flexibility by defining an artefact model with an explicit notion of structure in terms of a taxonomy. This taxonomy represents an abstract (simplified) self-contained view on the content of an artefact model. It then allows, for example, to couple the complex contents of an artefact model to the sub-models of a development process model (process integration), and it allows for the flexible application of the reference model at project level considering the selection and creation of selected artefacts in response to individual project characteristics.

REQ 2. *Explicit Representation of Domain Knowledge by defining a Concept Model*

Besides an explicit notion of structure, the artefact model needs to capture the concepts of the application domain in a concept model and, thus, explicitly represent the knowledge about an application domain.

The concept model thereby shall support awareness of producing syntactically consistent and complete results with a concept model that abstracts from domain-specific description techniques.

To support awareness in the creation of the results, the artefact model needs to conceptualise the domain's contents rather than operationalising this support by comprehensively defining conformance constraints. Such (executable) constraints would, e.g., support constructive quality assurance with respect to conformance in a tool-supported manner [Sch08a], which is, however, not in our scope.

In addition, we do not define the contents via a mathematical model in order to

1. support its application and interpretation by project participants without the necessary knowledge, and to
2. capture the functional and non-functional modelling concepts likewise (see also the discussion in Sect. 3.1.3.4).

However, it shall rely, where possible, on modelling theories to ensure the validity for the application domain (see also REQ 5).

REQ 3. *Support of Project Management by enabling Progress Control and by defining Responsibilities*

To effectively support project management, we need to support two major aspects: (1) progress control during the creation of the artefacts; (2) a concept for defining responsibilities in the creation of artefacts.

To enable progress control, we have to make use of two concepts. First, we need a status model that captures the actual degree of completion for an artefact (used in the context of project management and control). Second, to build on a status model, it demands in addition for means to describe the completion levels from an engineering point of view. For this, we need to characterise the concept model according to domain-specific levels of abstraction (see Sect. 2.2.3.2), as used in the area of model-based development. These levels of abstraction allow for the determination of whether artefacts are underspecified and the overall process therefore solution-oriented.

To enable the definition of responsibilities, we have to couple domain-specific roles to chosen artefacts according to the principles of artefact-based development process models, such as done in the V-Modell XT.

REQ 4. *Flexibility for Customisation by Modularity of the Meta Model*

Besides supporting the flexibility for customising a domain-specific reference model by means of a structural view on the artefact model (see REQ 1), the overall reference model and, thus, its meta model has to exhibit a modular structure.

With modularity, we consider the (logical) structuring of the meta model into the sub-models of a development process model (artefacts, roles, ...) and the explicit definition of associations between those sub-models. Both shall support the flexibility for customisation, e.g., regarding process integration of the reference model into a particular development process model (see also Kuhrmann [Kuh08b]). The explicit definition of the associations shall additionally support their (consistent) preservation during customisation.

REQ 5. *Validity for the Application Domain*

The last requirement considers not the meta model itself, but the domain-specific interpretation of the meta model. An interpretation is valid if (1) the domain-specific instance is described in the language of the meta model and if (2) the used concepts are valid for the application domain. In Sect. 3.3.3, we describe how to ensure the validity considering, e.g., the development and evaluation of the domain-specific instance of the meta model in an industrially hosted environment and the use of modelling theories to build the concept model.

3.2.2 Design of a Meta Model for Artefact Orientation according to Research Objectives

Figure 3.4 illustrates on the left side a sketch of the sub-models, which we define for our meta model for artefact orientation with respect to the ones generally used

3.2 Design of a Meta Model for Artefact Orientation

in development process models (def. in 2.5.1.2). These sub-models are defined in response to the requirements stated in the foregoing Sect. 3.2.1.2.

The right side of the figure depicts the requirements in relation to our particular design decisions. The arrangement itself into the modular structure of selected sub-models achieves (besides the associations between the sub-models) REQ 4.

As described in the foregoing section, REQ 5 considers the domain-specific interpretation of the meta model and is therefore not depicted in the figure.

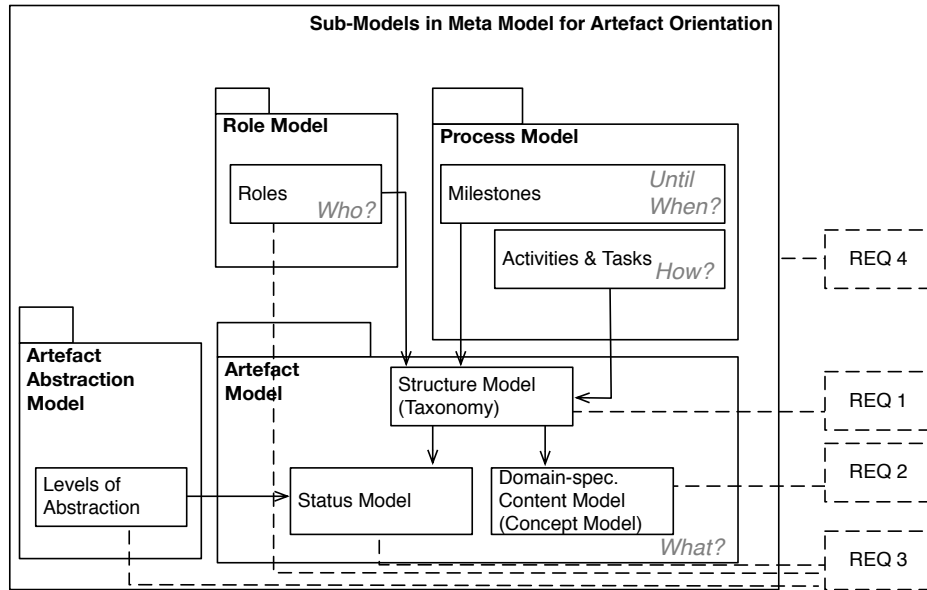


Figure 3.4: Sketch of meta model for artefact orientation according to requirements.

In the following, we give an overview of each of the depicted sub-models in relation to the previously stated requirements.

Artefact Model. For the representation of the artefact model, we perform a combination of two concrete variations, which by now are treated in literature in an isolated manner.

We define a structure model by means of a taxonomy. This structure model satisfies, with its coarse-grained view on an artefact, REQ 1 considering the flexibility for customisation (see also the discussion in Sect. 3.1.4). The further elements of a process (activities, milestones, roles) are then connected to the structural view, i.e., to artefacts and topics to be considered in the single artefacts.

Embedded into this structural view, we define a domain-specific content model by means of a concept model to satisfy REQ 2. The elements of the taxonomy then group selected sets of elements and relations of the complex concept model to support the flexibility (giving a simplified view onto the content).

The concept model itself represents the content view, i.e., it explicitly represents knowledge about the concepts of the application domain. Hence, it supports, at project level, awareness of creating artefacts in a syntactically consistent manner, but still allows for variations in the choice of a syntax. It offers an explicit notion of (content) structure to persist the concepts of the artefacts in documents or data sets. Note that the structure model is defined according to the concept model

(as it groups selected concepts) and, thus, it represents the structure of domain-specific documents and data sets.

Each artefact, respectively each element in the structure model, additionally includes a status model in order to support progress control (requirement 3).

Artefact Abstraction Model. We define an artefact abstraction model that defines domain-specific levels of abstraction for the content model, i.e., it relates the concepts in the content model to each other according to a domain-specific refinement hierarchy. The levels of abstraction satisfy together with the status model requirement 3.

Role Model. We define a role model and couple the roles to particular artefacts. These roles then take the responsibility of creating certain artefacts. Together with the status model and the levels of abstraction, the definition of responsibilities finally satisfies requirement 3.

Process Model. As introduced in Sect. 3.1.1, we follow in the definition of a process model the philosophy of artefact orientation. This means that we do not offer concepts to directly define a concrete process via interrelated activities and methods. The process is customised (enacted) via the definition of concrete milestones, which each is allocated to an artefact. The process then arises from a defined sequence of milestones and the actual creation of the artefacts respecting their (content) dependencies. In addition, the method structure has to be kept flat. That means that the methods must not be restricted to a particular description technique for representing each artefact, since this would barrier the flexibility for customisation (that demands for supporting variations in the creation of artefacts).

3.3 Meta Model for Artefact Orientation

In the following, we define the meta model for artefact orientation. This meta model covers the requirements defined the sections before.

We first give an overview of the meta model and the single sub-models, before defining each sub-model in detail. In the last Sect. 3.3.3, we conclude the chapter with a description of the validity procedure, which we performed during the instantiation of the meta model.

3.3.1 Overview of the Meta Model

Figure 3.5 illustrates the meta model for artefact orientation. The meta model is logically organised by means of packages. Each package represents a sub-model of the artefact-based approach, which is defined in response to our previously defined research objectives. We define:

1. An artefact model
2. An artefact abstraction model
3. A generic role model
4. A generic process model

In order to support the demanded modularity (REQ 4), we associate the elements between the different sub-models via specific *association classes*.

[illegible]

The definition of such association classes (and the chosen cardinalities) is inspired by the meta model of the V-Modell XT [TK09], because the V-Modell XT has a similar artefact-based philosophy and achieves our particular objectives regarding the flexibility for customisation. The association classes ensure the explicit definition of associations and their (consistent) preservation during customisation.

3.3.2 Sub-Models of the Meta Model

78

3.3.2.1 Artefact Model

As introduced in Sect. 3.2.2, we combine, as part of an artefact model, two self-contained artefact views, a structure model and a content model. The artefact model thereby offers two different views on the artefacts by incorporating an explicit notion of structure and an explicit definition of domain-specific concepts.

The structure model defines the structure of specification documents or data sets by means of artefacts being hierarchically decomposed into single content items. Each content item then represents containers for the concept model, e.g., chapters. The concept model then defines for the content items the concepts reflecting the elements and relations found in description techniques, as well as the (content-based) relations between the chapters.

With the structure model, we support a process integration of the reference model into a development process model, and the customisation of the model at project level. With the concept model, we support awareness of creating precise results.

Guiding questions for process integration are, for example: “What elements must be allocated to a chapter that includes a use case model? What methods and roles are allocated to that use case?” Guiding questions for awareness of precise results are, for example: “What elements are needed for describing a use case model and what syntax can be chosen for its representation?”.

The structure model and the content model are both organised by types. The types represent a naming convention for the elements in the development process for which project-specific exemplars can be created and should not be set equal to a type system for generalisation and specialisation (see also Sect. 2.5.2).

Artefact types can be instantiated to domain-specific documents or data sets, like a *Requirements Specification*. An artefact then represents a project-specific exemplar of this specification, such as a *Requirements Specification: Travel Ordering System*.

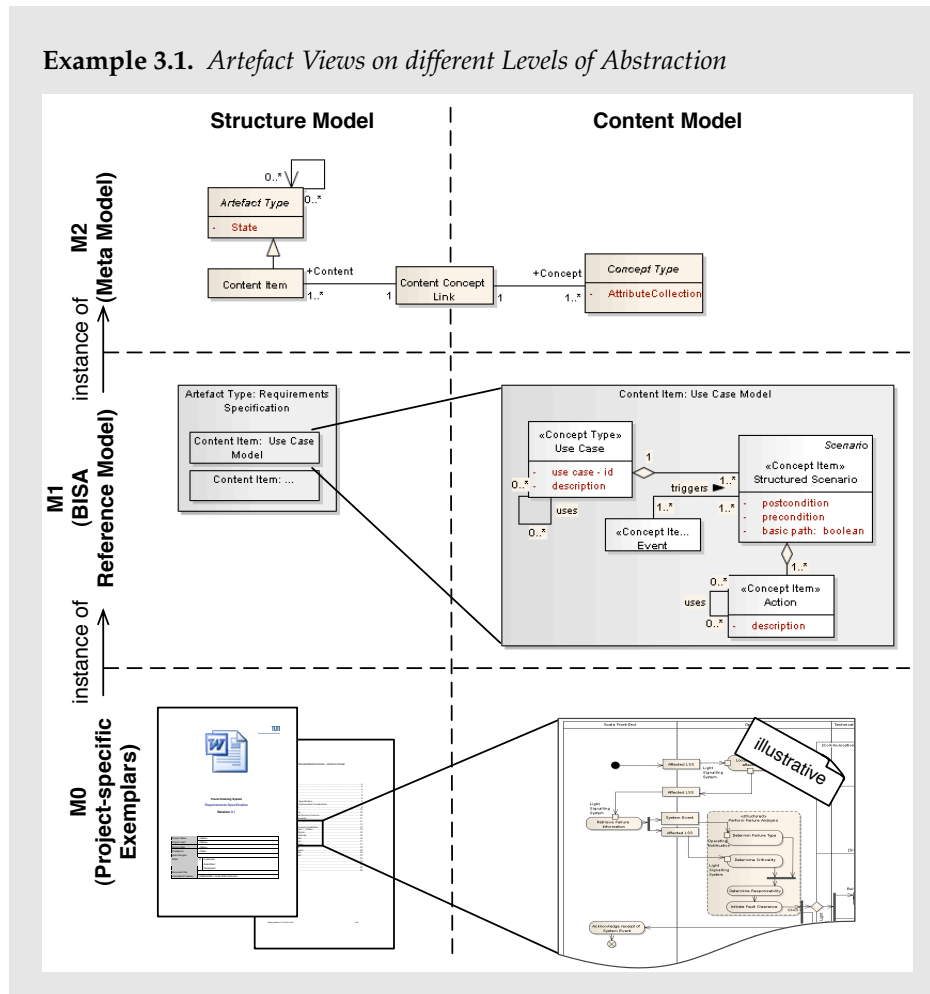
Regarding the artefact content, each artefact (type) is hierarchically organised by content items, like the content item *Use Case Model*. For each content item, we express its content by the corresponding excerpt of a concept model. Within this concept model, we refer to concept types, such as the concept type *Use Case*, and related concept items, such as *Scenario* with several *Actions*. A concrete concept then represents the project-specific exemplar, such as *UML Activity Diagram: Search Hotel*. The concept types *Use Case* can be described in different shapes choosing a specific syntax; for instance, by using an *UML Activity Diagram* in order to illustrate possible sequences of interactions (the scenarios) that are part of the use case. Another possibility could be to use *Message Sequence Charts*.

The different syntactic possibilities emphasise different aspects of the concepts in dependency to the intended assertion, but the instantiated concepts for the content item are still the same.

Example 3.1 illustrates, at the different levels of abstraction, the notion of artefacts to which we refer with our contributions (the meta model, the BISA reference model and the project-specific exemplars, see Sect. 2.5.2). In the figure, we explicitly separate for illustrative reasons the view on the structure of an artefact and the view on its content.

3.3 Meta Model for Artefact Orientation

Example 3.1. *Artefact Views on different Levels of Abstraction*



Finally, we define the term *artefact* in the context of the thesis as follows:

Definition: *Artefact*

An *artefact* is a deliverable that is produced, modified, or used by a sequence of tasks as part of one (development) phase and has value to a role. Artefacts are subject to version control and have a specific type³. They are hierarchically structured into content items that define single areas of responsibility and that are output of a single task. Each content item encompasses at its lowest level of decomposition:

1. **Concepts:** a concept defines the elements and their dependencies of domain-specific description techniques used to represent the concern of a content item. Concepts have a specific type and can be decomposed to concept items, in which the differentiation is made if different items of a concept can be described with different techniques⁴.
2. **Syntax:** the syntax defines a concrete language or representation that can be chosen for a specific concept.
3. **Method:** The method (respectively task) describes the sequence of steps that is performed in order to make use of a concept.

³ An artefact type represents a naming convention, e.g., "Requirements Specification" (see the example stated above).

⁴ See also example 3.1 on page 80.

Dependency Types. In the meta model, we distinguish two different types of dependencies: structure dependencies and content dependencies.

Structure Dependencies describe dependencies between different artefact types, mostly given by decomposition of artefact types into content items. Further dependencies can result from content-related dependencies; for instance, the dependency between the artefact types *Requirements Specification* and *System Specification*. Each structure dependency is defined as a direct outcome of the content dependencies, whereby the structure dependency in the meta model at hand corresponds, in part, to the content-related and the creational dependencies in the V-Modell XT (see also Sect. 2.5.3.2). The dependencies between the artefact types to further elements of the process model are, however, given by corresponding association classes in the generic process model.

Content Dependencies describe the dependencies between different concepts in the concept model; for instance, the dependency between a use case and an actor. Content dependencies are defined between the concept types in single artefact types or between the concept types of different artefact types. Compared to the dependencies illustrated in Fig. 2.3 in Sect. 2.2.3.3 introducing the dependency types in requirements engineering, the content dependencies correspond to the syntactic dependencies. In the mentioned figure, we additionally distinguish between inter- and intra-abstraction dependencies. In order to reduce the complexity in the meta model, we do not, however, additionally specialise the dependencies, because these arise as a direct consequence of the concept types being allocated to the levels of abstraction.

Conformance Constraints. Conformance constraints represent rules for the creation of dependencies in order to, e.g., support the syntactic completeness and consistency in a tool-supported manner. The conformance constraints apply thereby, if at all, to the existence of certain concept types and explicitly to the content dependencies between the single concept types.

Relation to the BISA Reference Model. The domain-specific artefact model is introduced in Chp. 4.3. In this section, we will give an overview of the structure model, before defining the concept model in subsequent sections. As part of the overview, we additionally define an overview of the used dependencies in Sect. 4.3.3, and over the artefact status model to support progress control.

The focus of the thesis lies on conceptualisation of artefact orientation for the purpose of customisation. Thus, as part of our contributions, we do not define domain-specific conformance constraints. To (manually) support the syntactic conformance to a certain level, we make use of UML class diagrams and describe the associations with the necessary cardinalities. Enriching the artefact model with comprehensive conformance constraints is part of future work, which is further discussed in Sect. 7.2.

3.3.2.2 Artefact Abstraction Model

The artefact abstraction model defines, as introduced in Sect. 2.2.3.2, vertical levels of abstraction and views.

Views. We distinguish two types of *Views*. *Modelling Views*, like structure or behaviour, depend on each other, but enable a structured description of systems and benefits an autonomous decision taking in which single aspects can be described

3.3 Meta Model for Artefact Orientation

without directly having to specify other aspects (see also Schätz et al. [SPHP02]). For example, a use case model can be described without directly having to describe participating actors in detail. *Business Domain Views* are additional means to structure specific subsets of artefacts according to, e.g., business processes. We make use of business domains to logically cluster large amounts of requirements in order to support the decision taking during customisation.

Vertical Abstraction. Regarding vertical abstraction, we describe *Levels of Abstraction*. The levels of abstraction define stages of refinement, over which artefacts are refined (and / or decomposed) for each of the modelling views [GW06, RJ01].

Relation to the BISA Reference Model. The domain-specific interpretation of the artefact abstraction model is presented in Sect. 4.2. In particular, we define in Sect. 4.2.1.2 the levels of abstraction while taking into account representative architecture frameworks. In Sect. 4.2.1.1, we define the modelling views. Based on the introduced levels of abstraction, we infer in Sect. 4.2.3 a definition of the terms *underspecified artefacts* and *solution orientation*.

3.3.2.3 Generic Role Model

The generic role model gives an abstract description of responsibilities that directly participate within, or indirectly contribute to the development process. As shown in the meta model on page 78, a role has responsibility of at least one artefact type and performs a set of activities in order to produce or modify the artefacts. This association is reflected in the association classes *Role Performs Activity* and *has Responsibility*.

Roles that indirectly participate within the development process may contribute to, or depend on the completion of an artefact, but they have no responsibility. By the genericness of the role model, it can be customised to individual needs, as the roles can be provided by the role structure of a development process model into which the reference model is integrated. The allocation of roles to project participants is performed as a particular customisation step.

Relation to the BISA Reference Model. We introduce the role model in Sect. 4.9.

3.3.2.4 Generic Process Model

The generic process model defines all entities necessary to define a specific process at project level.

In particular, we define *Milestones* that represent a points in time when an artefact must be completed. For this, artefacts (or content items) are coupled via the association class *Artefact Assigned to Milestone* to milestones. The order for the creation of the artefacts is directly defined by instantiating the milestones.

Further concepts are *Phases* and *Activities*. A phase (e.g. requirements engineering) defines a repeatable set of activities which, in turn, define major areas of concerns (e.g. requirements elicitation). There exist several *Tasks* for such an area of concern. Each task describes a sequence of atomic *Steps* that are performed by a role in order to produce, modify and / or use an artefact as input or output. More concrete, a task can be initiated by using an output of another task. It then describes the steps to read, modify or produce concept types with a selected syntax as an output.

Hence, a task is a method in the sense as it is defined in the area of method engineering. A task supports the selection of which concepts are needed for modifying or producing further artefacts by the choice of a particular syntax (see also Braun et al. [BWHW05]).

Relation to the BISA Reference Model. With the introduction of the phase *Business Information Systems Analysis*, we define the scope of our reference model in Sect. 4.1. In Sect. 4.8, we define the corresponding process model. The milestones are introduced in Sect. 4.8.3, while we define two activities with corresponding tasks considering the content creation in Sect. 4.8.4.

In contrast to the activity-based philosophy, which is reflected in method engineering [BWHW05], the tasks within the activities are defined in a generic manner and not restricted to a concrete syntax, since this would hamper the flexibility with additional restrictions. The restrictions on syntactic possibilities arise from the concept model that abstracts from existing description techniques. As long as chosen description techniques are used to exclusively express the elements defined in the concept model, they are suitable. See also Sect. 4.8.2 for further information.

3.3.3 Validity Procedure during Instantiation of the Meta Model

When instantiating the introduced meta model for a particular application domain, we have to ensure the validity of the instance (REQ 5, Sect. 3.2.1.2). This validity procedure is, however, twofold.

On the one hand, we have to ensure the (syntactic) conformance of the instance to our meta model. On the other hand, we have to ensure the validity of the instance for the chosen application domain, i.e., it must be suitable and cover the basic concepts of the application domain in a correct and sufficiently complete manner.

In the following, we describe the procedures, which ensure the validity of the BISA reference model of Chp. 4.

3.3.3.1 Conformance of the BISA Reference Model to the Meta Model

We ensure the conformance of the BISA reference model to the meta model by describing the reference model in the language formulated by the meta model. We use the concepts and relations defined by the meta model. We already showed which parts of the BISA reference model relate to the introduced sub-models of the meta model in Sect. 3.3.2. Since this thesis is focussed on conceptualisation of artefact orientation rather than on its (tool-supported) automatisisation, we refrain from using (operation) mechanisms as found in DSL-based modelling [GSCCK04].

3.3.3.2 Validity of the BISA Reference Model for the Application Domain

In order to ensure the validity of the BISA reference model for the application domain, we refer to the development procedure of our initial artefact-based reference model, on which the BISA reference model relies. This initial reference model was developed and continuously evaluated, in part, in an industrial context, hosted by Capgemini TS. To demonstrate that our contributions do not only satisfy the needs of this particular company, we change in the case studies in Chp. 6 the industrial context to another company.

In the following, we will exclusively focus on the validity of the BISA reference model for the application domain. The chosen development procedure, which ensures this validity in a constructive manner, is illustrated in Fig. 3.6.

3.3 Meta Model for Artefact Orientation

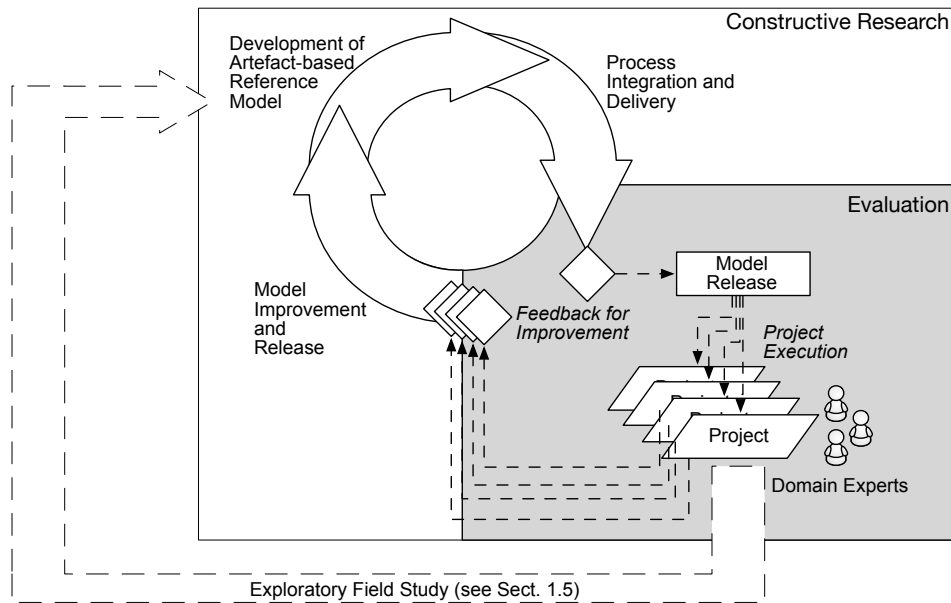


Figure 3.6: Development procedure of the artefact-based reference model.

The development procedure consists of three stages:

1. An *exploratory field study* as a preparation.
2. A *constructive research* in which the approach has been developed.
3. An *evaluation* in an industrial context.

The exploratory field study was performed once as a preparation. The actual development was performed in several iterations. These iterations included the development of the approach in an isolated manner and (after a process integration) its evaluation by domain experts.

In the following, we describe each of the performed stages in the development procedure.

Exploratory Field Study

We performed as a first step an exploratory field study [MFWLB10, MFWL⁺12]. We already introduced this field study and its study design as part of our research method in Sect. 1.5 (see the problem analysis).

As already shown, we performed the field study as a preparation for the development of the artefact-based reference model and the customisation approach. One particular aim of the field study was the determination of artefacts (respectively description techniques) that are of interest in different projects considering the domain of business information systems. We referred to similar or identical environments in which the reference model was evaluated while abstracting from specialities of the single projects given by individual project parameters.

Constructive Research

To avoid the development and especially the evaluation of the artefact-based reference model following a big bang strategy, we opted for an iterative procedure. The iterations included the development of the artefact-based reference model,

its process integration into the development process model of the company, and the delivery of the integrated approach to corresponding project environments for the purpose of evaluation. Based on the feedback from given domain experts, the model was afterwards improved and released as the company-wide reference model, before extending it in the next iterations.

The overall development took two years including four evaluations and releases, each performed every half year (in June and in December). Capgemini TS declared the reference model as the standard reference model for RE in custom software development projects after the fourth evaluation.

Please note that the final release of this standard reference model (*Quasar Requirements V.1.1* [MFC09]) covers after its process integration into the company-specific development process model not all aspects of the BISA reference model. In fact, the concept model of the *Requirements Specification*, of parts of the *Business Specification*, and of the *Traceability Matrix* are the same. Aspects of quasar requirements, which are defined according to the organisational culture (such as concrete methods) differ from the BISA reference model. However, the validity of the BISA reference model for the application domain is given if its concept model, which abstracts from domain-specific concepts, is valid. Hence, the validity, which we ensure by the development procedure depicted in Fig. 3.6, holds also for the BISA reference model.

Development of the Artefact-based Approach. Based on the findings of the study, we inferred the artefact model and the sub-models of the development process model, which are related to the artefacts. The inference of the reference model was performed by

1. identifying different description techniques, used to produce the artefacts, and establishing the relations between the techniques from a methodological point of view (considering a refinement hierarchy),
2. abstracting from the identified description techniques to deploy the concept model,
3. harmonising the concept model with the levels of abstraction considering the refinement hierarchies of the envisioned application domain,
4. inferring and coupling methods considering the creation of each concept type with all concept items, and
5. inferring the structure model according to the concept model and allocating milestones to the identified artefact types.

Considering step two, one major problem was that many available description techniques offer no or a weak semantic foundation of the underlying concepts. This affected the correctness and completeness of the concept model (see the discussion in Sect. 3.1.3.4 and Broy et al. [BFH⁺10]). Hence, we took into account modelling theories, wherever possible, and additionally generalised specialities of description techniques that are developed as specific-purpose concepts for particular methods (see REQ 5 on page 75). The different description techniques were reduced to their least common denominators and brought in line with corresponding modelling theories, if available.

Turner [Thu04] defines, for example, a formalised view on the concepts of business processes using the *FOCUS* theory [BS01]. In her approach, she introduces different concepts and relations including the ones of “Roles” that participate in the execution of business processes. The *Business Process Model and Notation* (BPMN) [Bus03], in turn, introduces (methodologically motivated) two concepts for the description of roles: “Pools” and “Lanes”. According to the modelling theory, we abstract, however, from these two roles concepts. Therefore, according to

3.3 Meta Model for Artefact Orientation

our understanding of the purpose of models stated in Sect. 2.5.1.1, the defined concept model simplifies specialities in order to (1) reduce the complexity of the model and (2) ensure a sufficient correctness considering its applicability with different notions.

Regarding step three, we had to consider that many concepts depend on domain-specific levels of abstraction, which are not taken into account in the corresponding description techniques. For instance, approaches for specifying services can refer to means for describing the state of the envisioned system, e.g., via state machines. The description of states depends, in turn, on the actual level of abstraction. If we use, for example, the concepts for describing single services considering the system as a black box, the inclusion of (complex) state machines for each of the described service is questionable, until the services are not decomposed and not set in relation to each other (as part of a component architecture within a glass box view on the system). Hence, the levels of abstraction in the envisioned application domain had to be harmonised with the defined concept model and vice-versa.

Process Integration and Delivery. After developing parts of the artefact-based reference model, it had to be evaluated. This evaluation aimed at getting feedback from domain experts participating in real life projects for further model improvements. We performed a process integration of the concept model into the development process model of the company (a company-specific deviate of RUP) in order to prepare the evaluation. Afterwards, we internally distributed the document with the current release of the integrated reference model with instructions for the feedback (including spreadsheets for documenting the review and a deadline for the feedback).

Model Improvement and Release. We collected the feedback from the domain experts over a specific time frame. Each review comment was prioritised by the domain experts as

- *mandatory for the next release*, or as
- *mandatory, but not for the next release*, or as
- *optionally*.

In case of conflicts, e.g., by contradictory comments of some reviewers, we notified these reviewers and organised a workshop for discussing and resolving the conflicts. We then improved the model according to the review comments. If comments considered exclusively individual aspects of the project environments in which the domain experts participated, we marked these aspects as such in the reference model. For instance, if a domain expert, participating in projects in the telecommunication sector, demanded for a content item “Feature Specification”, we highlighted this content item to be relevant for this sector.

After improving the reference model according to the comments, we delivered the updated reference model to all project participants for acceptance. They could accept the reference model, conditionally accept it, or deny it. After the acceptance of all domain experts, we internally released the reference model with release notes, which indicated to the differences to the previous releases and to planned extensions of the reference model in the next release.

Evaluation

In the following, we describe the performed evaluations. We have, however, to stick to an abstract description of the evaluations for confidentiality reasons.

The evaluations were performed by domain experts of the company. We considered three company-specific roles: *Project Lead*, *Business Analyst*, and *Chief Architect*. The project lead was responsible for the overall project, the business analyst for the business process and the requirements specifications, and the chief architect for the logical component architecture of the system.

The domain experts reviewed the delivered reference model with respect to their experiences in their own project environments in which they actually participated or in which they participated in the past. These project environments are the ones that were in scope of the previously performed field study and are summarised in Tab. 3.2.

Table 3.2: Project environments in which the evaluation was performed.

Project Environment	Industrial Sector	Project's Effort
P1	Finance	Small
P2	Finance	Small
P3	Finance	Small
P4	Retail sale	Medium
P5	Contracting authority	Medium
P6	Telecommunication	Large
P7	Logistics	Large
P8	Logistics	Large
P9	Aerospace	Medium
P10	Contracting authority	Medium
P11	Finance	Medium
P12	Automotive	Large

At the beginning of the development (in the field study and in the evaluations), we referred in total to 16 different project environments, but reduced them to 12, because the domain experts were not available anymore or because they couldn't give access to project-specific information for confidentiality reasons.

In order to ensure the independence of specific industrial sectors and project-specific parameters, the project environments were distributed over different industrial sectors and had different (planned) effort. The effort is approximated to three categories: up to 20 person years as small-scale projects; from 20 to 120 person years as medium-scale projects; above 120 person years as large-scale projects. Except for the projects P1 to P3, the projects were performed in collaboration with external parties and most of the project environments were distributed over different European countries⁵.

Finally, besides the structured evaluation procedure by the domain experts and their reviews, we considered further projects as pilot projects. In these projects, we gave training courses and supported setting up the tool infrastructure. We accompanied the projects with direct support and additionally took their comments into account for further model improvements.

⁵ With the distribution of project environments, we refer to the locations of customers and contractors.

3.3 Meta Model for Artefact Orientation

CHAPTER 4

Artefact-based Reference Model for Business Information Systems Analysis

In the previous chapter, we have analysed the principles of artefact orientation and inferred according to our objectives a meta model for the philosophy. In this chapter, we present a domain-specific interpretation of this meta model for RE in the domain of business information systems. After introducing the phase *business information systems analysis* (BISA) in Sect. 4.1, we define for each sub-model of the meta model the domain-specific instance. In Sect. 4.2, we describe the artefact abstraction model. We relate a set of basic concepts for modelling behaviour and quality aspects and infer a definition of when artefacts are under-specified and the process therefore solution-oriented. In Sect. 4.3, we introduce the artefact types and describe each in detail in subsequent sections including a description of the interfaces for a process integration in Sect. 4.7. In Sect. 4.8, we define the process model, before concluding with a description of the roles in Sect. 4.9. The content of this chapter is based, in part, on our contributions in [MFK09, WMFIL09, LWMFB10, IHMFJ09].

At the end of this chapter, the reader will have a comprehensive overview of the single sub-models of an artefact-based RE approach covering the needs of the envisioned application domain. Based on this view, we define, in Chp. 5, the artefact-based customisation approach considering process integration and the project-specific creation of the artefacts with respect to individual project characteristics.

Contents

4.1	Business Information Systems Analysis	90
4.2	Artefact Abstraction Model	91
4.3	Artefact Model Overview	105
4.4	Artefact Type: Business Specification	111
4.5	Artefact Type: Requirements Specification	126
4.6	Artefact Type: Traceability Matrix	141
4.7	Artefacts as Interfaces for a Process Integration	142
4.8	Process Model	143
4.9	Role Model	147

4.1 Business Information Systems Analysis

In Sect. 2.4, we introduced a first domain-specific interpretation of RE. For this, we illustrated, at the example of the architecture framework IAF, to what extent RE covers domain-specific levels of abstraction. This coverage was inferred from our understanding of the principle “IT business alignment” that corresponds to the principles of RE aiming at an iterative approach for problem statement and problem solving. At the example of the domain-specific levels of abstraction, we then argued that RE should cover in the application domain the description of

1. the current and the future state of business processes, and of
2. resulting constraints towards particular information systems.

According to this interpretation, we now introduce the phase *Business Information Systems Analysis* (BISA). BISA, as the domain-specific interpretation of RE, covers activities and artefacts used to describe needs related to the business (processes) of a company, and ones related to single information systems.

Figure 4.1 illustrates the activities and the artefacts to which we refer in BISA. We put them in relation to areas of levels of abstraction in the architecture framework, similar as done in Sect. 2.4, where we described the interrelations between RE and the application domain (Fig. 2.6 on page 33).

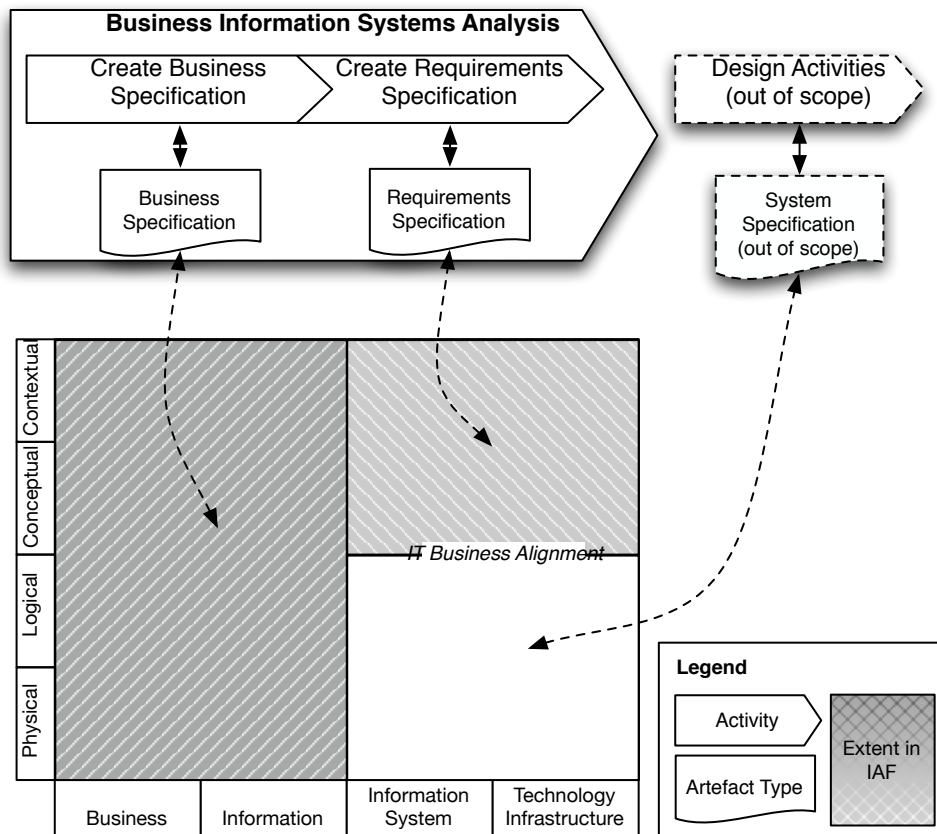


Figure 4.1: Scope of the phase *Business Information Systems Analysis* (BISA).

For the domain-specific interpretation of RE by means of the phase BISA, we now distinguish in the framework two areas. The first (business) area that is depicted on the left side of the figure, and the area considering resulting demands towards

single systems (and their development process), depicted on the upper right side of the figure.

The concepts used to describe the business processes are incorporated by the artefact type *Business Specification*. The concepts used to describe demands towards single systems are incorporated by the artefact type *Requirements Specification* (being aligned with the content of the business specification).

For the domain-specific interpretation of RE, we distinguish between two activities and two artefact types, instead of simply referring to requirements engineering and one corresponding artefact type. The reason is that RE, in general, emphasises more the needs towards systems and their development process rather than towards a business (see [GW06] and Sect. 2.2.3.2 introducing the levels of abstraction referred by RE approaches). The (re-)design and description of a business architecture is, however, not in scope of RE. Such concerns are known in the application domain as *business engineering* [Thu04], which covers an own discipline with domain-specific methods and description techniques. Development process models used in the application domain, such as the *Rational Unified Process* [JBR99, KK03], thus include (besides RE) business engineering, respectively “Business Modelling”, as a self-contained discipline. RE, in turn, is meant to contribute approaches to infer from the output of business engineering the description of demands towards the systems (and the development process).

Hence, we define according to this domain-specific interpretation the phase BISA as follows:

Definition: *Business Information Systems Analysis*

The phase *Business Information Systems Analysis (BISA)* comprises two activities for an iterative and systematic creation of the two artefact types:

1. *Business specification* reflecting the needs of all relevant stakeholders towards the current and the future state of the business.
 2. *Requirements specification* reflecting the needs and constraints towards information systems and their development, aligned with the business needs stated in the business specification.
-

In subsequent sections, we refer to this interpretation and define the sub-models of the meta model defined in Chp. 3.

4.2 Artefact Abstraction Model

In the following section, we describe the artefact abstraction model. According to the meta model in Sect. 3.2.1.2, the artefact abstraction model serves

1. to understand the domain-specific levels of abstraction over which artefacts can be refined and / or decomposed and, thus,
2. to enable progress control by means of completion levels and, finally,
3. to enable the determination of whether the (content of) artefacts remain underspecified and the process therefore is solution-oriented.

We subsequently discuss the levels of abstraction and the modelling views, which we consider as relevant in the context of BISA. Afterwards, we describe how the levels of abstraction relate to selected concept types in Sect. 4.2.2, before concluding with a definition of solution orientation in Sect. 4.2.3.

4.2 Artefact Abstraction Model

4.2.1 Levels of Abstraction and Modelling Views

Figure 4.2 illustrates the levels of abstraction over which (BISA) artefacts' contents are refined and / or decomposed and the modelling views that separate, within each level, concepts according to their concern (see also the meta model in Sect. 3.3.2.2).

The levels of abstraction result from the views taken by domain-specific architecture frameworks for SOA and our domain-specific interpretation of RE. For this, we depicted in Fig. 4.1 two major areas within an exemplary architecture framework (IAF). One area is considered by the business specification and another area is considered by the requirements specification. The differentiation between those two areas aims at aligning the needs of a business with needs towards a set of information systems.

		Modelling Views			
		Environment	Behaviour	Data	Structure
Levels of Abstraction	Organisation's Context	What is the company's strategic orientation and the underlying principles? What are the company's steering objectives?			
	Business Process Hierarchy	How is the business structured and what capabilities must the business internally and externally offer to satisfy the objectives?			
	Business Process Logic	How are the capabilities realised, what information is processed and how is the (business process) work flow realised?			
	Information System Service Hierarchy	What functionalities shall information systems offer to support and (partially) automatise the business capabilities? How shall single information systems be used and what expectations have future users towards the system in the context of their business processes and further (external) systems?			
	Information System's Constraints	What are resulting logical and technical restrictions towards the systems, their properties and their architecture to fit the intended use?			

Legend

Extent of the Business Specification

Extent of the Requirements Specification

Figure 4.2: Levels of abstraction and modelling views considered in the BISA reference model.

To define the (vertical) levels of abstraction in the artefact abstraction model, we

combine the aspect areas and the levels of abstraction (“Architectural Layers”) given by the IAF to a hierarchy of levels. We take into account the architecture model of the Technische Universität München [BFG⁺08], which gives a simplified view onto system-relevant levels of abstraction, and related work in RE, which describes requirements-relevant levels of abstraction (see Sect. 2.2.3.2).

We consider for BISA in total five levels of abstraction. The first three levels organise the concepts used to describe the business needs, e.g., the information about the (idealised) business processes of a company to be supported by information systems.

The fourth and fifth level of abstraction organise the concepts used to describe the resulting requirements towards information systems being aligned with the business specification. These levels are defined according to the requirements-relevant levels introduced in Sect. 2.2.3.2. We explicitly consider with the term *requirement* for the sake of clarity requirements towards the information systems and restrictions on the development process (see also Sect. 4.5 discussing requirements concepts). In the following, we describe the modelling views and the levels of abstraction.

4.2.1.1 Modelling Views

We define four modelling views, which group the concepts of the application domain. Since many concepts arise from informal description techniques and / or have cross-cutting concerns by nature (such as it is the case for non-functional concepts), we first define the modelling view considering behaviour modelling, e.g., as found in available description techniques for modelling business processes (see for example [Thu04, Bus03]). We then extend this view with further three views, which are used to separate concepts that indirectly relate to behaviour modelling. The four resulting views are as follows:

1. *Behaviour*, grouping the concepts used to describe functionality (and work flows), such as a sequence of process steps within a business process model or a sequence of actions that a user performs when interacting with an information system.
2. *Structure*, grouping the concepts used to structure the behaviour by means of, e.g., business domains or components, if considering a logical component architecture.
3. *Data*, grouping the concepts used to describe the entities that are processed or interchanged by a piece of behaviour; for instance, information (system) objects that are referred in business process models or use case models.
4. *Environment*, seen in a broader sense, as a means to group the remaining concepts that describe the contextual aspects such as ones that are of interest for behaviour modelling; for instance, roles that participate in the business processes, goals that shall be satisfied by the execution of business processes, or external systems.

The business domain views, introduced in Sect. 3.3.2.2 (used to logically group requirements), are not described in this section, since we introduce the necessary concepts in Sect. 4.4.

4.2.1.2 Levels of Abstraction

We consider five levels of abstraction and introduce them according to Fig. 4.2 from top to bottom. Each level of abstraction encompasses concepts that have to be satisfied by the concepts of the following level, independently of the order

4.2 Artefact Abstraction Model

in which the concepts are created. Exemplary concepts and their relations to the levels are introduced in the next section.

Organisation's Context. The organisation's context defines long-term steering principles and high-level objectives of a particular company, such as the targeted market and how this market shall be incised.

Business Process Hierarchy. The business process hierarchy describes a structured taxonomy of internally and externally offered business capabilities that satisfy the organisation's context¹. The business capabilities are described via services without giving details of their internal realisation by means of a concrete processes. Such a work flow description of the work that is carried out in order to provide the service is given by the next level of abstraction. Thus, this level can be seen as a black box description of business processes. We describe, for example, the business service "Course Administration" and "User Administration" and in which business unit the corresponding processes are performed. Please note that the duality of the notations for modelling services and processes is introduced in Sect. 4.2.2.1.

Business Process Logic. The business process logic describes the internal realisation of the business services, i.e., process steps performed in order to provide the services. Hence, this level gives a glass box view on the (business) behaviour. According to Thurner [Thu04], we see the business process logic to take an operational and internal perspective on the work flow and to thus consider all concepts that are used to describe such work flow, without taking into account particular information systems. For example, we describe the process steps that are performed within the business process "Administrate Courses" (like the steps performed when adding new courses), which roles perform the steps, and what data is communicated as part of the steps.

Information System Service Hierarchy. The information system service hierarchy describes the demands on systems' functionalities necessary to support chosen process steps. This level includes descriptions of how users will interact with a system in the context of their business processes, while the system is seen as a black box. The user-centric view on the (at the systems' borders) externally visible behaviour of the systems gives no details on its internal realisation in terms of a set of interacting logical components. For instance, we describe, by means of use cases, how users interact with the system when adding new courses. The concepts used in this level represent in general the user requirements, if referring to the approach of Wieggers [Wie03]. If referring to Gorschek et al. [GW06], the level corresponds to the function level (see also Sect. 2.2.3.2).

Information System's Constraints. The information system's constraints describe logical and technical restrictions on a system's architecture, its functionality by means of single atomic actions, and its quality by means of assessable system quality requirements (see also Sect. 4.2.2.2). If referring to Wieggers' approach [Wie03], this level includes the system requirements. We consider concepts that describe the transition to logical and technical architecture layers. Although including quantified restrictions and implications towards a system's architecture,

¹ We do not distinguish externally and internally offered services, e.g., services offered to a market and ones offered to internal departments of a company.

we explicitly do not take into account its internal realisation with logical components. We describe, for example, single actions that the system performs when a user adds new courses (e.g., “Search Trainer”), or actions that the system performs when interacting with external systems (e.g., “Verify Time Schedule”). Hence, at this level of abstraction, we see a system as a grey box rather than as a glass box, since we restrict systems’ internals, but do not consider their logical structure by interacting components, interface specifications, and functions. However, many requirements can arise from a methodological point of view as a consequence of design activities in which component interactions are defined.

With the introduced levels of abstraction, we can define a hierarchy of concepts to describe the problem space. Further levels of abstraction are not in scope of the thesis. We depict, however, in appendix B.2.4, the dependencies between chosen concepts to (generic) concepts for modelling a logical component architecture.

Discussion in Context of RE-specific Levels of Abstraction

In Sect. 2.2.3.2, we introduced levels of abstraction considered by contributions in the area of RE. Some interrelations of related work with the introduced levels have already been explained. The requirements abstraction model by Gorschek et al. [GW06], however, proposes two levels of abstraction that have not been considered: a goal level, respectively product level, and a feature level. We do not consider the feature level, since features relate to a set of requirements concepts that can be stated, in turn, at different levels of abstraction. Goals are means of making statements of intents and are decomposed over different levels of abstraction. Hence, we allocate neither features nor goals to an explicit level of abstraction, but relate them to selected concepts that are used over different levels (such as goals influencing business processes).

4.2.2 Basic Concepts and their Decomposition and Refinement

In the following, we give an overview of how the concept types in the artefact model relate to the artefact abstraction model introduced above. As a first step, we give an overview of an exemplary set of functional concepts and then of how refinement relates to concepts for describing quality aspects.

Based on this understanding, we can define a notion of underspecified artefacts and infer a definition of *solution orientation* in Sect. 4.2.3.

4.2.2.1 Introduction into Basic Concept Types

We distinguish between concepts for modelling functional behaviour and non-functional modelling concepts. Concepts for modelling functional behaviour build the backbone of our artefact model, since these concepts directly rely on a notion of refinement. We refer, in particular, to a set of concepts as they have been introduced in Sect. 2.3.1.2 in the context of SOA:

1. Business processes and business services, structured via business domains.
2. Use cases, information system services, and functional requirements, used to describe the system-supported realisation of the processes.

The first ones are specified in the business specification, the second ones in the requirements specification.

4.2 Artefact Abstraction Model

In direct relation to those concepts, we define concepts for modelling non-functional aspects as they directly relate via intra-abstraction dependencies to functional concepts. For instance, the concept for describing actors that relate to the concepts of a use case model, are situated at the same level of abstraction as the use cases themselves. Non-functional concepts thereby relate to different levels of abstraction while their (inter-abstraction) dependencies have not a notion of refinement, but of, for example, decomposition or realisation.

In Fig. 4.3, we depict an exemplary set of concept types according to the levels of abstraction and the modelling views introduced in the foregoing section.

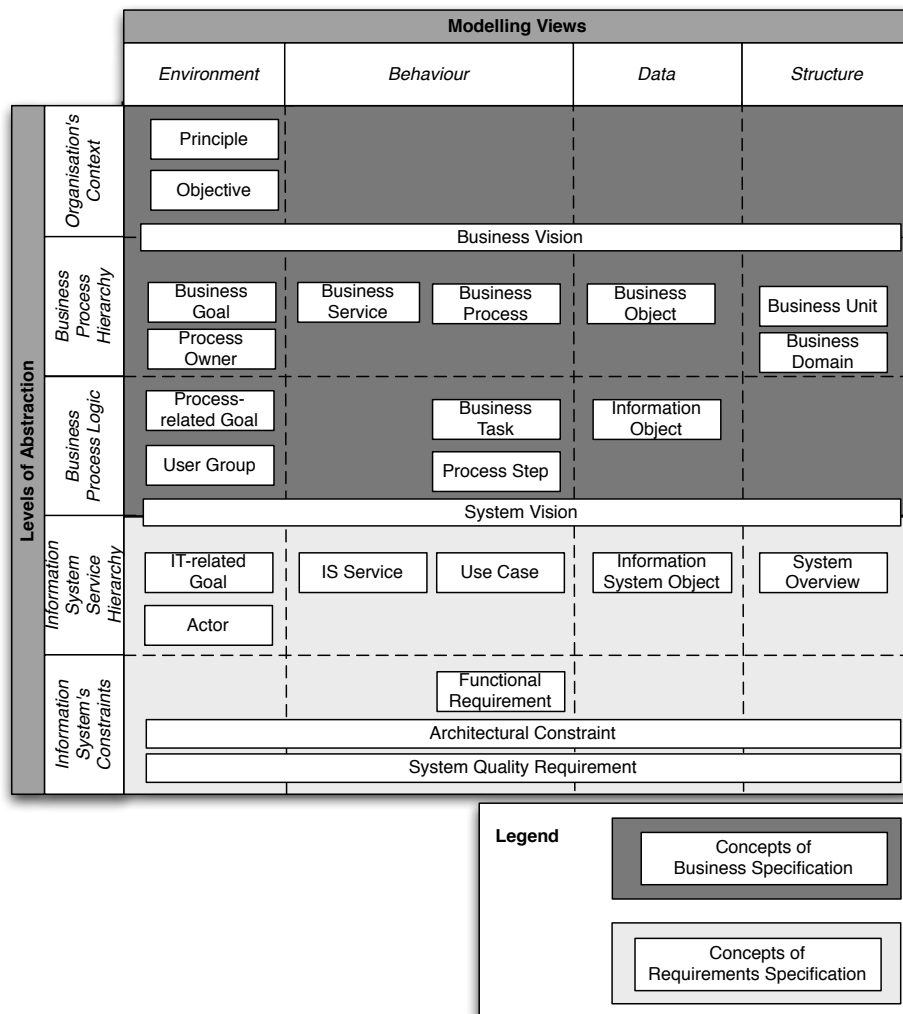


Figure 4.3: Concept types in relation to artefact abstraction model.

In the following, we introduce the depicted concepts from top to bottom according to the artefact types by which they are included. A particular focus lies on the concepts for modelling behaviour on the different levels of abstraction.

Concepts of Business Specification. Motivated by a set of long-term *Objectives*, we form (business) behaviour via *Business Processes* and *Business Services* in

direct relation to corresponding *Process Owners* that take the responsibility for the processes. Both the processes and the services are means to describe behavioural properties, i.e., work that is carried out in a company, but the two concepts take different views on this behaviour at different levels of abstraction.

In our context, a service defines a piece of behaviour that can be decomposed to atomic steps, while the single steps, if put in relation to each other, form a particular process [BLMF⁺10]. From a modelling perspective, we use business services as an explicit means to abstract from the details that are defined as part of a business process model. The latter, in turn, defines a set of activities that are performed in order to provide the behaviour.

This means that we use services to give a black box view onto single pieces of behaviour (e.g., “Skiing Course Management”), while the business processes refine this behaviour into a concrete work flow (glass box). This work flow is represented by a set of business tasks (e.g., “Create Seasonal Course Plan” or “Book Skiing Instructor”) that can be further decomposed to *Process Steps*. Those process steps represent atomic (not further decomposable) activities used to form the actual work flow. This work flow then describes an abstraction of all possible instances of the activities that are performed by single *User Groups* in order to provide the business behaviour (e.g., the steps “Generate new Course” followed by “Check Availability of Skiing Instructor”). As part of those steps, we also specify the *Information Objects* being processed, e.g., a “Course Attendee”.

The process steps then are the ones that are potentially supported by functionality of an information system, i.e., they are the ones that potentially are *realised* by system actions (e.g., as part of a use case model). From a theoretical point of view, the process steps represent again self-contained (atomic) services. From a conceptual point of view, we refer to explicit modelling concepts. Further information is given in Sect. 4.4.2.2.

Regarding non-functional concepts, we make use of *Business Domains* to structure behaviour models, and *Goals*. Business domains group logically related business processes for which one process owner is responsible, e.g., “Course Administration” grouping all business processes being in scope of the process owner “Course Manager”. Goals form prescriptive statements of intent used to argue for certain (business) activities. For this, we refer to the goal-driven modelling approach *Knowledge Acquisition in Automated Specification of Software* (KAOS) [vL03, vL04]. However, we conceptually distinguish different levels of abstraction over which we decompose those goals (see also [EHH⁺08] classifying goals according to levels of abstraction). We decompose *Business Goals*, like “Expand Market”, to *Process-related Goals*, like “Decrease Process Lead-time”, to motivate the content in the process models. We specify, for example, which single steps a skiing instructor exactly performs when teaching a skiing course and what particular process-related goal he shall achieve.

Concepts of Requirements Specification. In the requirements specification, we conceptualise how information systems shall realise the specified business processes. For this, we make use of *Use Cases*, *Information System Services*, and *Functional Requirements*. These functional modelling concepts form a basis to describe what process steps shall be realised in interaction with an information system. In context of the functional models, we also describe what information objects shall be reproduced by the system by means of *Information System Objects* (e.g., which information (data) is necessary when booking a skiing course in interaction with a system).

The modelling of system functionality at the information system service hierar-

4.2 Artefact Abstraction Model

chy can be performed in two complementary ways, both used to specify system-supported behaviour in relation to the system-independent one captured in the business specification. We distinguish *Information System Services* and *Use Cases*. Both concepts are means to describe (black box) system behaviour.

Use Cases describe sequences of interaction between *Actors* (*realising* user groups) and the system as a whole. More precisely, a use case represents a collection of interaction scenarios, each defining a set of interrelated actions that either are executed by an actor or by the system under consideration [Coc00]. We will refer to an action performed by an actor as an *Actor Action*, and to an action performed by the system as a *System Action*. Now, for modelling how the business process shall be supported by a system, we use each of the specified process steps as a candidate to be system-supported or to remain as an activity to be performed by an (external) actor. Each process step thus is realised either by one system action or by one actor action, whereby the transition from the business process models represents a *realisation*-relation between the concept of the process step and the actions in the use case scenarios.

Services, instead, describe a logical representation of a use case, not necessarily involving actors or concrete sequences of interaction. An information system service thereby represents user-visible functions that the system shall offer and is described via input/output-relations [HT09, Rit08], i.e., the mapping of (typed) inputs and outputs, both represented as information system objects.

Functional Requirements form an additional concept that is used to describe actions in the interaction between an (external) actor and the system, i.e., one actor action followed by one system action w.r.t. a stated condition (see also Davis [Dav93]). From a theoretical point of view, functional requirements are atomic services and, thus, actions in a use case scenario. However, functional requirements are explicitly provided by conceptually different means [BLMF⁺10].

Furthergoing information on functional modelling concepts used in the requirements specification is given in Sect. 4.5.2.2.

The actual system specification (the component architecture) is not in scope of our approach. However, we now introduce the simplified relation of the introduced concepts to the *Function* concept used when specifying a system architecture. Schätz defines a function as “a capsule of system behaviour, defined by an external interface in terms of data and control flow” [Sch08b]. Whereas a function is provided via ports (interfaces), it conceptually *realises* atomic services, respectively system actions in a use cases scenario. Hence, we use a function as a means to explicitly represent the realisation of user-centric requirements as part of the (internal) specification of system functionality, similarly as done for the transition from the description of system-independent process steps to system-supported ones defined via use cases. See also appendix B.2.4 for further information.

Please note that the cardinalities between the functional modelling concepts, e.g., between a function and a system action, or between the actions in use cases and process steps of a process model, are only 1:1 if seen “ideally”. The cardinalities depend on the methodologically chosen granularity of the specified behaviour models, or of the component architecture what, in turn, is still an unsolved issue.

Business and System Vision. For a manageable set of specification documents, we additionally make use of concepts for scoping. With scoping, we refer to defining relevant subsets of concepts of one level of abstraction, before entering the next. Before specifying the concepts of the business process hierarchy in full, we define a *Business Vision*. Before specifying detailed requirements, we define the scope of the system within the *System Vision*.

Both content items are means to formulate initial ideas about a development project and the system under consideration aligned with all relevant stakeholders. For example, within the system vision, we capture the relations between the system actions in the use cases and the process steps, and define the borders of the systems in relation to external systems. Hence, the visions support the transition from the organisation's context to the following business levels and from the business process logic to the information system service hierarchy. For this reason, we also couple milestones to the content items that comprehend the visions (see Sect. 4.8.3).

4.2.2.2 Refinement and Decomposition of Quality-related Concepts

So far, we discussed refinement of concepts for modelling functional behaviour. In this section, we illustrate similar principles for the refinement and decomposition of quality-related concepts.

Quality is, in general, a multifaceted topic, since there exist different views onto the term *Quality* with a diversity of interpretations [Gar84, KP96], and finally no commonly accepted definition [Gli07]. In the area of RE, there exist approaches that describe the decomposition of quality characteristics (like "Security") in parallel to functional needs to detailed requirements while both areas are handled in an isolated manner. These approaches propose the combination of both quality requirements and functional ones at their last level of abstraction; for instance, by describing quality requirements as part of use cases that they affect, as proposed by Doerr et al. [DKVKP03, DKK⁺05] or in REM [BPKR09, BGK⁺07, GBB⁺06]. Such a decomposition and combination with use cases, however, is problematic, because

1. quality requirements have a cross-cutting nature affecting not only use cases (e.g., as it is the case for maintainability requirements),
2. the decomposition and refinement depends on the application domain and the corresponding levels of abstraction, and because
3. all concepts used to express customer's needs, including the ones of a use case, imply a notion of quality.

A business process that involves information systems in use, for example, has a particular value to a business. This can be understood as quality, too. Thus, use cases themselves can be seen as a means to express quality.

For a domain-specific interpretation of *Quality* and *Quality Requirements*, we make use of quality definition models (see also Deissenböck et al. [DJLW09]). Using a quality model supports not only a unified terminological agreement, but also an integrated view on quality requirements and, e.g., quality assurance. We refer to the *Activity-Based Quality Model* (ABQM) of the Technische Universität München [Dei09], because of the activity-centric nature of the concepts used for BISA with, e.g., business processes and use cases. The ABQM is based on early effort of Boehm et al. [BBK⁺78] and McCall et al. [CM78], and supports the integration of quality-related concepts into the levels of abstraction at hand.

The activity-based quality model defines quality via a set of system properties and their associations to (business) activities carried out during the use of the system [Dei09, DWP⁺07]. For example, quality can be expressed by the structuredness of a system architecture and its positive effect on maintenance activities that an administrator performs. The externally perceived quality, respectively the "Quality in Use" (as stated by the *ISO Std. 9126 for Product Quality* [ISO03] and the currently developed revision *ISO/IEC FDIS 25010* [ISO10]), is then reflected in the supported business activities. The resulting restrictions on the properties of information systems, necessary to enable the support of the activities, is named

4.2 Artefact Abstraction Model

in our context *System Quality Requirements*, which we allocate to the last level of abstraction. We already showed the integration of the activity-based quality model into the RE refinement hierarchy and evaluated its use in different case studies [WMFIL09, LWMFB10, WDW08].

According to the principles defined by the quality model and our previous work, we now conceptually integrate different concepts for modelling quality as part of the introduced levels of abstraction (see Fig. 4.4). According to Cockburn [Coc00], we decompose goals over different levels of abstraction to motivate the refinement of behaviour, respectively the decomposition of business activities and system interaction scenarios, and to infer (assessable) system quality requirements being aligned with the high-level business needs.

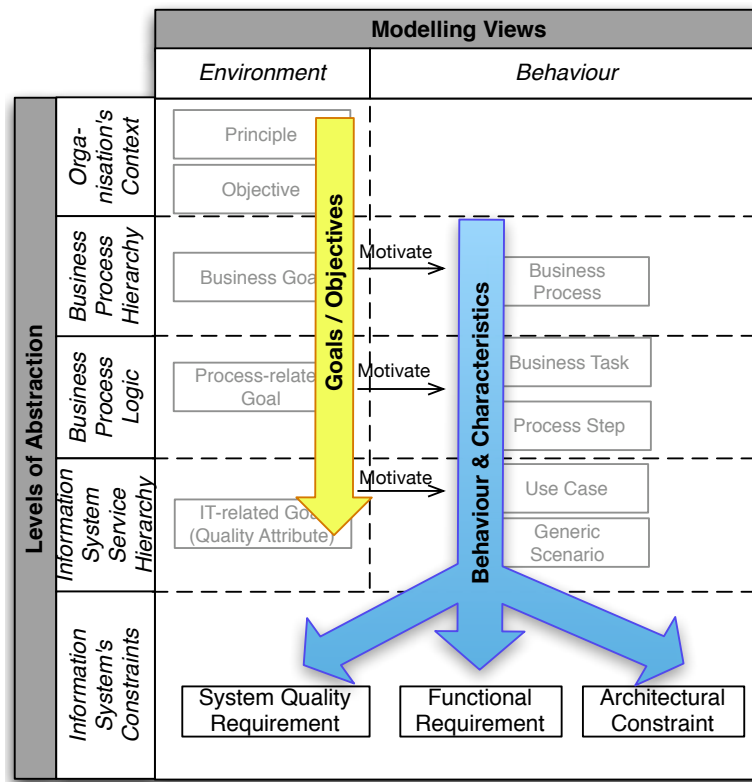


Figure 4.4: Quality-related concepts in relation to artefact abstraction model.

The decomposition of goals goes from high-level business goals, over process-related goals that steer the specification of activities, to IT-related goals. The latter represent quality attributes to be fulfilled by a system as found in the ISO standards for software quality (see also [Boe08]), such as “Security” or “Maintainability”.

The behavioural view defines structural aspects of the business, internal work flow realisations, and finally how information systems are used in the context of business processes (see also the foregoing section). Since the introduced use cases are functional means that describe system interactions in the context of business processes, we extend this functionally motivated view with generic scenarios. This concept describes sequences of interactions with a system, similarly as it is the case for use cases, but is not restricted to the context of business process logic. Generic scenarios involve interaction scenarios including all its system’s facets (documentation, code, ...); for instance, attack scenarios that shall be prevented (known as “Misuse Cases” [Ale03, HP08]), or activities of an administrator, such as code

modifications.

Having this comprehensive view on activities that can generally be performed by the use of information systems, we can infer at the last level of abstraction quantified, or at least assessable, requirements. These (system) requirements describe restrictions on the properties of a system in relation to the stated activities to be supported. We classify the used concepts according to Glinz [Gli05, Gli07] depending on the different possible concerns towards system's details into

- functional requirements, describing single actions a system shall perform,
- system quality requirements, constraining a system and its environment in its properties and conditions by means of metrics and values, or
- architectural constraints, logically and technically restricting the system in its architecture (orthogonal to both requirements types), regardless of the effects on the system's quality.

Other requirements types, having an organisational nature, are introduced as part of the artefact model.

4.2.3 Solution-Oriented and Problem-Oriented Refinement

We already showed in the fundamentals in Sect. 2.2.1.1 that the process, in which we make use of the introduced concept, is cyclic by nature [AW05b], reflected also in the chosen definition of business information systems analysis in Sect 4.1. Thus, with each of the made design decisions, the problem space is narrowed down for further requirements [CW97], since it is questionable to completely formulate the requirements in detail without continuously having a progressing idea of a future system in mind [GBB⁺06, BPKR09].

A major problem, however, arises from an uncontrollable solution-oriented processes in which the requirements are dominated by solution ideas, i.e., discussed on the basis of solution ideas while remaining underspecified (see the problem statement in Sect. 1.1). Although this problem is recognised, there still exists little guidance in literature due to the missing understanding and agreement on *solution orientation*, or more precisely on when artefacts remain *underspecified* causing a solution-oriented process. Such terms can be found in many contributions, but they are mostly associated with specific-purpose approaches; for instance, as done by Doerr et al. [DKVKP03] supporting a solution-oriented process by means of a knowledge repository, or by Pohl [Poh07] who states that requirements that are underspecified imply risks of misinterpretations. In fact, a definition of *solution orientation* and of measurements for the determination of underspecified artefacts causing solution orientation must rely on a refinement hierarchy and associated completion levels for the corresponding artefact types (see also Sect. 3.3.2.2 discussing progress control in artefact orientation).

Based on the introduced levels of abstraction and the followed principles of refinement and decomposition of the artefacts' concepts, we can now give such a domain-specific interpretation.

For this interpretation, we refer to the results of the field study introduced in Sect. 1.5, and transfer the discovered effects of solution orientation on the artefacts' completeness to the artefact abstraction model at hand. Taking into account this domain-specific interpretation of solution orientation, we define in Sect. 5.3 the customisation approach.

4.2 Artefact Abstraction Model

4.2.3.1 General Effects of Solution Orientation on Artefacts' Completeness

In Chp. 1, we introduced a field study performed as a preparation for the contributions of the thesis. One particular investigation were different *RE execution strategies* that were identified on the basis of so-called *artefact patterns*, i.e., clusters of specified content items (of different investigated projects), which exhibit similarities in their degree of completeness. We investigated (via k-means cluster analysis) the differences between the found execution strategies, reflected in the differing completeness of the documented content items. Thus, we were able to discover probable execution strategies while abstracting from variations in the performed process.

Table 4.1 summarises an excerpt of the found content items that have differing values between the resulting artefact patterns. We identified as a whole three execution strategies, but take only two into account, because one execution strategy arises from specialities of a change management procedure of the particular company. A detailed explanation of the cluster analysis and the results is given in [MFWL⁺12].

Table 4.1: Completeness of selected content items in different execution strategies found in practice [MFWL⁺12].

Content Item (REM)	Solution Orientation	Problem Orientation
Business Objectives *	●	●
Risk Calculation	●	●
System Success Factors **	○	●
Application Scenarios **	●	●
User Interface **	○	●
System Boundaries **	●	●
Assumptions	○	●
Traceability: Business to Req.	○	●
Traceability: Req to Sys.Spec.	○	●

Legend: ● Completely specified content item
 ● Incompletely specified content item
 ○ Missing content item
 * Reflects (business) behaviour models in BISA
 ** Reflects external system behaviour models in BISA

The summarised set of content items (clusters) in relation to the two artefact patterns causing either a problem-oriented or a solution-oriented execution strategy result from the artefact model of REM [GBB⁺06] that we took as a reference for the analyses (see also Sect. 3.1.3.2). Since the content items of REM are defined independent of an application domain, we now explain the two execution strategies according to the concepts and the terminology introduced in foregoing sections to support in the following section a BISA-specific interpretation.

We take into account that the content item “Application Scenarios” represents in REM all previously introduced concepts for defining behaviour and makes no difference between business process models and interaction scenarios, e.g., use cases. Thus, we took during the study the content item “Business Objectives” as a means to express business process models. In the table, we depict this mapping of the domain-independent content item to the concepts used in BISA with the symbol “*”. We use “**” to express which content items reflect the concepts used to define the external behaviour of information systems.

Problem Orientation. The analysed projects that followed this execution strategy had a profound specification of business processes with clear work flow descriptions. A similar degree of completeness is given in the external system behaviour, including a specification of the system boundary and (external) quality aspects. The strong focus on the problem space is also reflected in the traceability that includes the linkage between the elements in the requirements specification and the elements in the business specification, as well as between the elements in the requirements specification and the ones in the system specifications.

Solution Orientation. Projects following this execution strategy have a weak specification of functional behaviour models affecting also non-functional concepts (system boundaries, ...). Concrete work flow descriptions in the business process models were missing, same as functional behaviour models of information systems, e.g., expressed via use cases and functional requirements. Requirements were directly discussed on the basis of developed solution ideas, such as high-level architecture drafts, and remained at the level of abstract feature lists or service descriptions. These incompletely specified content items also hampered traceability.

4.2.3.2 BISA-specific Interpretation of Solution and Problem Orientation

As shown in the foregoing section, solution orientation implies incompleteness in detailed behaviour models that describe the processes and / or interaction scenarios with (atomic) functional requirements.

An interpretation, in terms of the introduced levels of abstraction, leads to the finding that a solution-oriented process hampers the creation of concepts that give insights into detailed behaviour

- at the business process logic, and / or
- at the information system's constraints.

Since the concepts used to express quality rely on the same refinement hierarchy (see Sect. 4.2.2.2), we can generalise the findings to levels of abstraction rather than defining certain execution strategies by their effects on single content items.

We thus use the term *underspecified* as follows:

Definition: Underspecified Artefacts

A *content item* is underspecified if the created concepts and relations are in a state being not conformant to the corresponding concept types, i.e., *incomplete*.

An *artefact* is underspecified if it has underspecified content items and these do not underlie further refinement, i.e., if the created concepts remain at the level of:

- the *business process hierarchy* in case of the business specification and / or
 - the *information system service hierarchy* in case of the requirements specification.
-

A solution-oriented process thereby results from underspecified artefacts as it is illustrated by Fig. 4.5. The figure depicts the levels of abstraction and how the different execution strategies relate to those levels.

In case of a problem-oriented process, the refinement is a continuous one in which each concept of a particular level of abstraction is completely constructed and has its rationale in a concept of a foregoing level — regardless of any order in which the concepts are created.

4.2 Artefact Abstraction Model

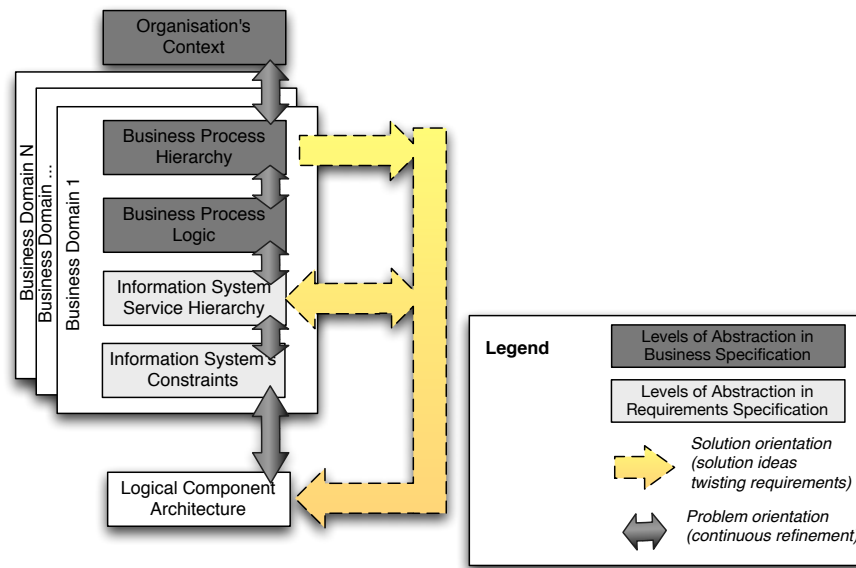


Figure 4.5: Interpretation of problem-oriented refinement and solution-oriented refinement in the context of the BISA reference model.

The problem-oriented strategy is depicted by the dark arrows and defined as follows:

Definition: Problem Orientation

Problem orientation considers the strategy of a continuous approval process in which the artefacts are analysed, refined, and quantified in consistency and compliance to the growing logical architecture.

Instead, and depicted by the dotted (yellow) arrows, solution orientation describes the strategy in which solution ideas dominate the stated problems.

In particular, the concepts at the information system service hierarchy would be absorbed according to a gained understanding of the logical architecture, which then would be designed without further quantification of the requirements to the last level of abstraction. Similarly, the business process logic would be designed according to the (architecture) solution, instead of vice-versa. The outcome would be discussions and negotiations with stakeholders based on an architecture-driven solution at the cost of refining information to the level including work flow descriptions and / or assessable (atomic) requirements that both are not worked out (see also Sect. 4.2.2 introducing different concepts for modelling same behavioural aspects on different levels of abstractions).

Accordingly, we define the solution-oriented strategy as follows:

Definition: Solution Orientation

Solution orientation considers a strategy in which artefacts remain during the process underspecified. They are directly distorted by solution ideas as an outcome of design activities. The concepts that describe the problem space remain incomplete at the level of the business process hierarchy and / or the information system service hierarchy.

Due to the risks of incomplete artefacts, a pure solution orientation is questionable. At the same time, a pure problem orientation is also questionable. The appropriateness in the degree of completeness (the balanced problem orientation) varies from project to project and with respect to the content from business domain to business domain in dependency to individual project parameters. Supporting this appropriateness is in scope of the customisation approach in Chp. 5.

4.3 Artefact Model Overview

In the following, we give an overview of the artefact types and how we represent each artefact type in subsequent sections. We discuss dependencies used in the structure model and in the content model, and conclude this section with an overview of the artefact status model.

4.3.1 Overview of Artefact Types

In foregoing sections, we already gave a first impression of the artefact types and selected concepts to which we refer in the context of the levels of abstraction for BISA. We distinguish in total three artefact types: the *Business Specification*, the *Requirements Specification*, and the *Traceability Matrix*. The structure and the content of each artefact type results from the development procedure described in Sect. 3.3.3:

1. We identified in the field study the domain-relevant artefacts, respectively description techniques (see Sect. 1.5).
2. We built the concepts by abstracting from those domain-specific description techniques, integrating them on the basis of a unified model, and associating the resulting concept types according to the artefact abstraction model.
3. We inferred the structure model according to the concept model.

Figure 4.6 illustrates an overview of the three resulting artefact types. For each of the artefact types, we additionally depict the top-level content items and omit, in a first step, for reasons of complexity further (decomposed) items.

In Sect. 4.4, we introduce the business specification in detail. This artefact type comprises all goals (statements of intent), capabilities (behaviour), restrictions, and conditions that affect the business of a company considering its current and future state.

In Sect. 4.5, we introduce the requirements specification. This artefact type comprises all demanded properties of information systems, organisational restrictions on the development process, and restrictions on the integration, that each shall be accomplished by a development project in satisfaction with the content of the business specification.

Finally, in Sect. 4.6, we introduce the traceability matrix, which comprises the (project-specific) associations between concepts of the business specification and concepts of the requirements specification.

4.3.2 Representation of Artefact Types in the Thesis

Each of the artefact types, depicted in Fig. 4.6, is described in subsequent sections according to the elements defined in the meta model (see Sect. 3.3):

1. *Structure Model*: We define via a taxonomy the artefact structure and give for each content item a brief description. Since the content items can vary during customisation in their arrangement, we additionally introduce in

4.3 Artefact Model Overview

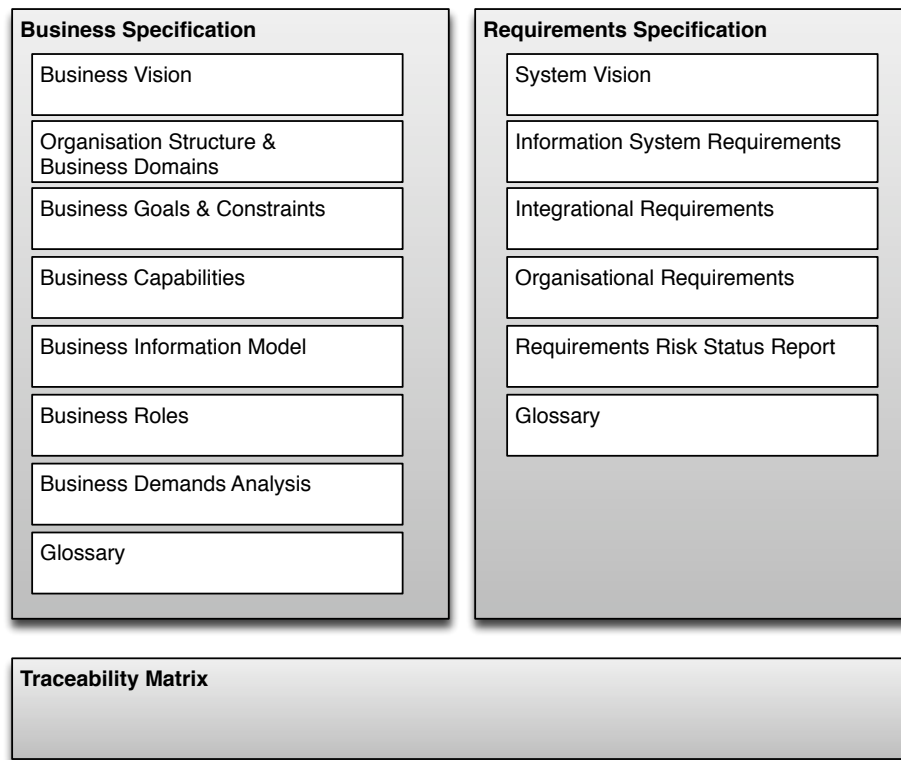


Figure 4.6: Overview of artefact types including top-level view on content items.

appendix B.1 different possibilities for arranging the structure of project-specific artefacts.

2. *Content Model:* We define in relation to the previously introduced content items the concept model and, thus, the content of the single items (and the relation to other content items). We organise the concept model according to the different modelling views. To represent the associations between the content items and the concept types (the *Content Concept Link*), we embed the concept model into the defined taxonomy and use for both models stereotypes, which we colour in different grey scales. For each concept type, we introduce and discuss related work from which the concept types abstract.
3. *Syntax:* We illustrate syntactic possibilities for representing the introduced artefact types.

Wherever reasonable, we illustrate for the creation of the concepts an excerpt of the running example introduced in Sect. 2.7. This example considers the re-design of business processes and the displacement of the (ski) course management system in the fictitious company *Alpine Adventure Tours* (AAT).

Finally, we stick in the thesis to the definition of selected concepts, as they have been introduced in the artefact abstraction model in Sect. 4.2.2. In appendix B.2, we illustrate the comprehensive concept model and supplement it with definitions within the glossary in appendix A. A detailed representation and discussion of the overall artefact model is also given in our previous contribution [MFK09].

4.3.3 Overview of Dependencies in the Artefact Model

As described in Sect. 3.3.2.1, we distinguish in the artefact model structure dependencies and content dependencies (see Fig. 4.7)

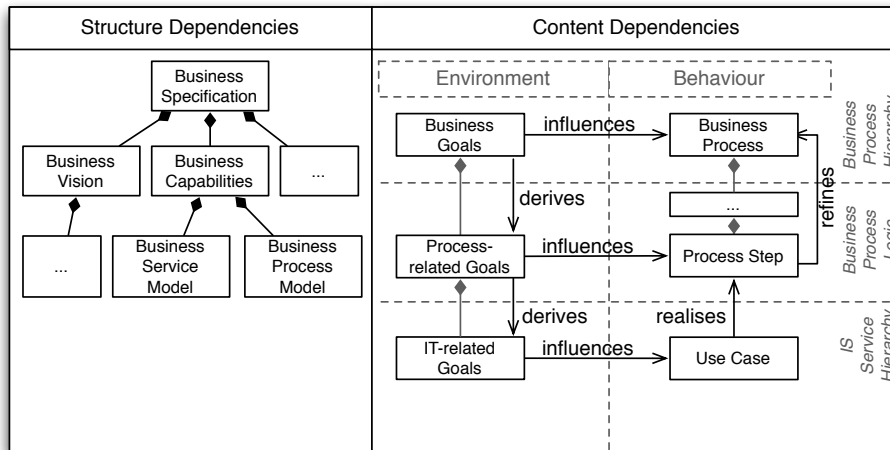


Figure 4.7: Overview of dependencies in the BISA artefact model (simplified).

On the left side of the figure, we represent the structure dependencies, which are considered by the structure model. On the right side, we depict the domain-specific interpretation of the content dependencies, which are considered by the content model. Both dependency types are introduced in the following.

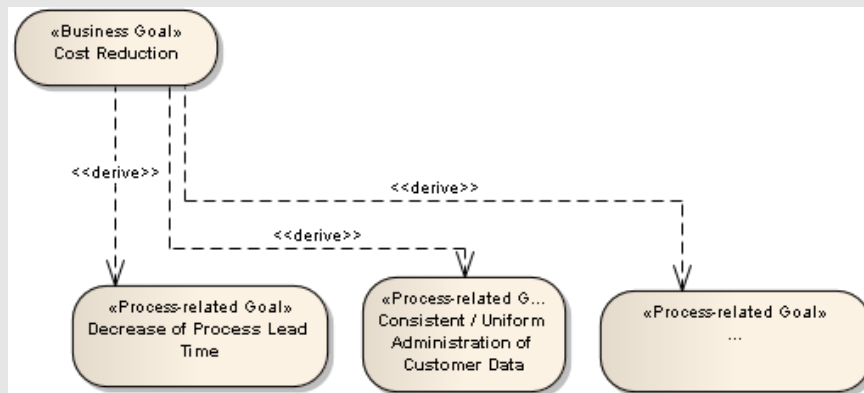
Structure Dependencies. Due to the hierarchical nature of the taxonomy in the structure model, we refer in the structure dependencies to *decomposition* (illustrated with the solid diamond shape). Some of the decomposed content items additionally have relations to other content items resulting from the content dependencies (like influences). Since these dependencies are covered by the content model, we do not capture these explicitly in the structure model to reduce the complexity.

Content Dependencies. In the concept model, we refer to a variety of dependencies and depict the most frequently used ones in Fig. 4.7. Following in the figure the inter-abstraction dependencies from top to bottom, we also make here use of *decomposition*. For instance, goals are decomposed over several levels of abstraction, similarly as done when decomposing business processes. We capture, however, in addition to decomposition further (semantic) dependencies. According to chosen description techniques, from which we will abstract, we make those dependencies explicit. We refer, in particular, to *derives*, *influences*, *refines*, and *realises*. Please note that we depict the dependencies in Fig. 4.7 for illustrative reasons in a simplified way and introduce them in detail as part of the concept model.

Derives means that if deriving concept B from concept A, B's assertion is defined on the basis of A (see example 4.1). If B is derived from A, B automatically satisfies A, but not necessarily vice-versa. This satisfaction is made, where necessary, explicit with the *influences* dependency.

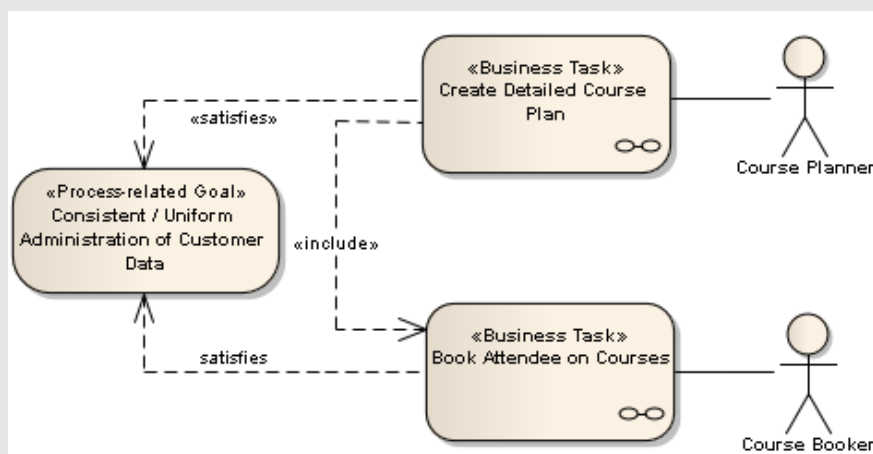
4.3 Artefact Model Overview

Example 4.1. Derives-Dependency at the Example of Goals

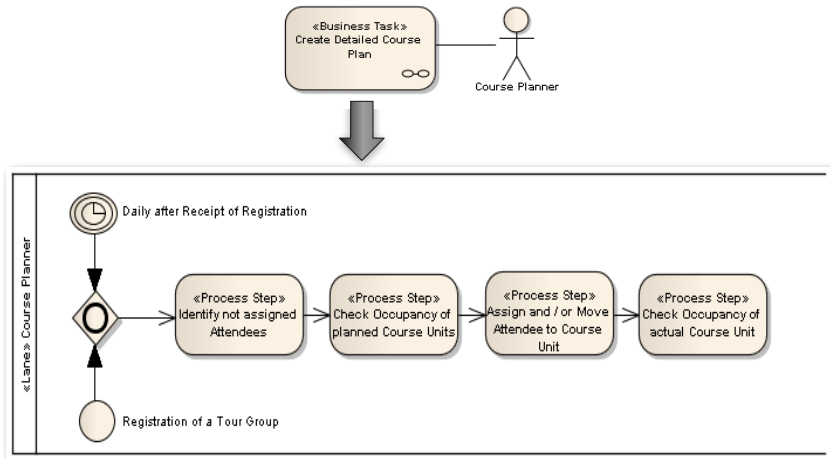


We refer to *influences* as a means to express that the assertions of two concepts can either support each other (we speak then of *satisfies*) or that they can be conflicted, i.e., they can make contradictory assertions. For example, a business process can satisfy a goal, which is achieved by the execution of that business process (see example 4.2). Furthermore, if A satisfies B, A might not have been necessarily derived from B.

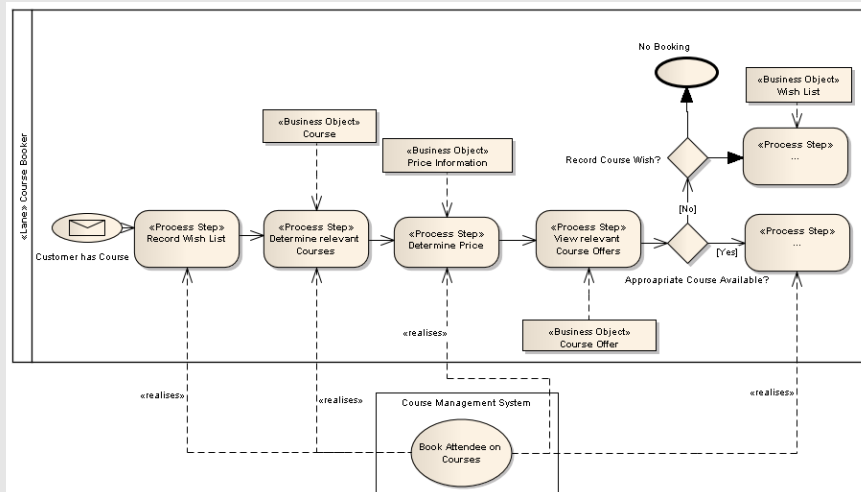
Example 4.2. Influences-Dependency (“Satisfies”) at the Example of Processes



While decomposition and derivation mostly refer to concepts with non-functional character, we additionally refer with *refines* to refinement of concepts for behaviour modelling over different levels of abstraction (see also Sect. 2.2.3.2). We take again an example with two concepts A and B, while B represents the glass box view on A. If an assertion made via A is then refined to B, B is a proper subset of A, enriched with more information. This also means that A can be reproduced on the basis of all assertions to which the one of A has been refined. For instance, a concrete business process can be reproduced on the basis of all the process steps to which the business process has been decomposed (see example 4.3).

Example 4.3. *Refinement at the Example of Processes*

Finally, we refer in the concept model to *realises*. A realisation describes that the assertion made with one concept is, at least in part, reproduced by further concepts. We refer to this dependency in order to express the transition from the content of the business specification to the content of the requirements specification (in order to describe how information systems shall *realise* selected business needs). As depicted in example 4.4, a use case, respectively the actions of the scenarios encompassed by a use case, realise selected process steps in the work flow description of a business process².

Example 4.4. *Realises-Dependency at the Example of Processes and Use Cases*

In contrast to refinement, the assertion of the process steps cannot necessarily be reproduced by the use cases, since these only include selected parts of the pro-

² For reasons of complexity we depict in the figure the realises dependencies between process steps and overall use cases instead of depicting the dependencies between process steps and actions in single scenarios and / or actors of a use cases.

4.3 Artefact Model Overview

cess. We could, for instance, decide that the verification of customer data shall be realised by another system while this realisation is not in scope of our analysis.

In addition, we refer with *realises* not only to concepts in the behavioural modelling view, but also to, e.g., dependencies between concepts in the data view.

4.3.4 Artefact Status Model for Progress Control

We define the artefact status model according to the one of the V-Modell XT (see Sect. 2.5.3.2) where each instantiated artefact type is subject to progress control. As illustrated in Fig. 4.8 with an UML statechart, we distinguish three stages after the creation of an artefact:

1. An artefact can be *in process*, i.e., it is elaborated and modified during BISA activities.
2. If no (external) quality assurance is assigned, the artefact can be *finished*, after it is successfully self-checked by the responsible role.
3. If external quality assurance is necessary, the artefact is *submitted* to the like. In this case, the artefact is checked according to selected quality attributes by independent roles; for example, as part of a quality gate by means of, e.g., an inspection. This aspect covers also an acceptance of external parties like the customer. Such a check can be successful or it can fail. If it fails, the artefact needs further modification and thereby returns to the state *in process*. Otherwise, the artefact is approved and changes its state to *finished*.

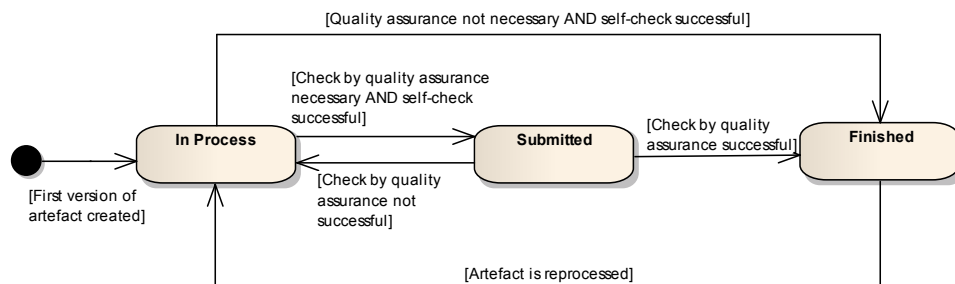


Figure 4.8: Artefact status model for progress control of artefacts.

We refer to these states during the operationalisation of the artefact model by means of, e.g., the process model. The progress control enables, for example, the determination of when the defined milestones have been reached (see also Sect. 4.8.3).

Possibilities for Extension. The introduced status model is intentionally kept simple in order to offer possibilities for extensions. These extensions can be performed in two ways, while both depend on the development process model into which the artefact-based approach is integrated.

First, the status model can be applied to single content items rather than to overall artefact types. Although this extension increases the effort for progress control, it might be useful to enable the initiation of critical activities that rely on single content items; for instance, to support the creation of a bid proposal, which is based on selected content items of the requirements specification. Second, it is also possible to extend the states of the status model; for instance, by defining additional states like *In Revision* resulting from particular needs of quality assurance.

An example for such an extension is the one from Katz [Kat09], which is based on our initial artefact-based approach (see also Sect. 3.3.3 introducing this company-specific artefact model and its relation to the contribution of the thesis).

4.4 Artefact Type: Business Specification

So far, we introduced the artefact abstraction model and the main ingredients of the artefact model. In this section, we describe the particular artefact type *Business Specification* following the structure introduced in Sect. 4.3.2.

4.4.1 Structure Model

Figure 4.9 illustrates the standard structure of the business specification. We intentionally define with this taxonomy the most abstract view that can be taken on project-specific exemplars of the artefact type. Each of the depicted associations between the content items relies on a 1:1-dependency, i.e., the business specification consists of one content item for the business vision, one for the organisation structure and the business domains, and so on.

Regarding the project-specific exemplars, we have different possibilities for structuring the content by means of additional, multiple content items. For instance, we can structure the content item *Business Process Model* by means of additional content items for each of the business process, such as “Business Process: Create Course” or “Business Process: Create Customer”. In appendix B.1.1, we illustrate different variations of the structure model being of interest at project level.

In the following, we introduce the single content items of the business specification, before describing the used concepts.

Content Item “Business Vision”. The business vision introduces company-specific problems and defines the resulting scope of the business that shall underlie a change as part of a (development) project. The business vision steers the stakeholders into a common direction by summarising aimed objectives (needs) to be achieved, principles to be preserved, affected business capabilities to be gained and / or changed, and resulting (re-) structures of envisioned organisations. For this, we refer to the following content items.

The *Business Context* characterises the envisioned company by including a description of the industrial sector, the incised market, and resulting cornerstones of a corporate philosophy by means of the *Business Drivers, Objectives, and Mission Statement*, and the *Principles* (subsequently defined according to [IIB09, Gro06a]).

The *Business Drivers, Objectives, and Mission Statement* includes a description of

- business drivers, describing internal and external factors, including individuals, knowledge and conditions, that initiate and support the design of business activities,
- business objectives, describing statements of intents, which concern major achievements of an organisation (typically within specific time frames), and
- the business mission describing a statement of direction and goal for the overall business.

The *Principle* includes restricting directives, respectively statements of belief, approach or intent, which serves as a basis for directing the formulation of (business) architectures.

4.4 Artefact Type: Business Specification

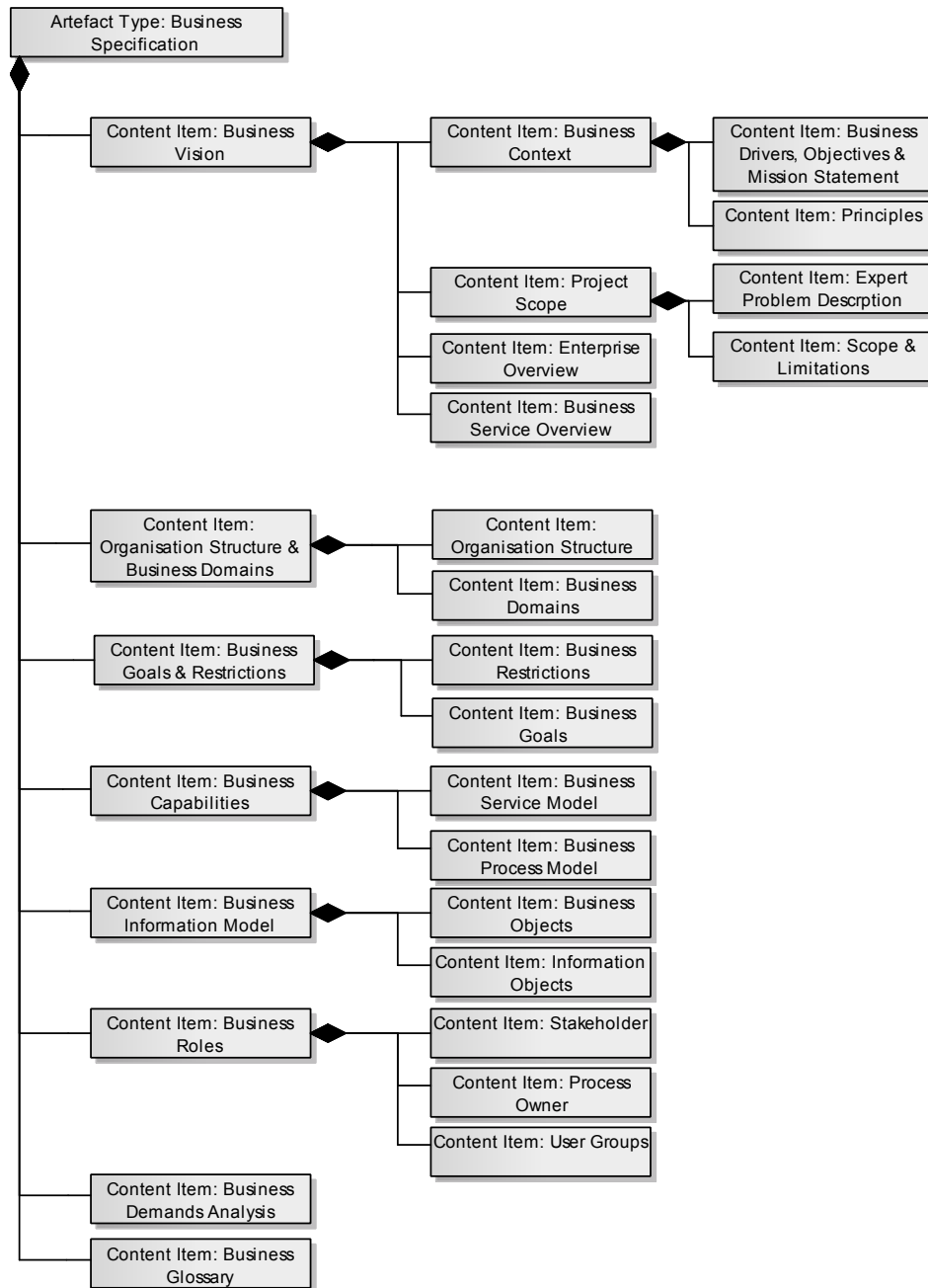


Figure 4.9: Structure model for the artefact type *Business Specification*.

We formulate with the *Project Scope* a description of the actual problems of a business and its information systems, and a conclusion with the resulting objectives of a project. To this end, we refer to

- the *Expert Problem Description*, which informally describes from the perspective of a domain expert (potential) problems that arise from the current state of the business, and
- the *Scope and Limitations*, which describes the scope of the project necessary to tackle the identified problems and limitations for the project.

With the *Enterprise Overview*, we summarise (possibly different) organisations and included business units that are affected by a project. According to [IIB09], we see an enterprise as a set of organisations and their business units that share a set of common business goals and collaborate to provide specific products or business services to customers.

Finally, the *Business Service Overview* summarises the business services of a customer that are affected by the project. It includes existing business services being displaced or modified and new business services that will be realised. Each business service is described in detail within the business service model.

Content Item “Organisation Structure & Business Domains”. We formulate two content item. First, an overview of the *Organisation Structure* that hierarchically organises the relevant business units and “determines key reporting lines and in conjunction with a cultural analysis will indicate where key stakeholders are positioned” [Gro06a]. We see an organisation as an autonomous set of hierarchically structured business units with defined boundaries, and a business unit as a recognised part of an organisation under management of at least one stakeholder. Second, an overview of the *Business Domains* that define, regardless of the organisation structure, decomposable sets of logically related business processes being related to one process owner (see also [EHH⁺08]). Business domains formulate the structure for business processes from a functional perspective following the principal of high cohesion, while potentially cross-cutting several business units. Because business domains structure the content of the business specification, as well as the one of the requirements specification, they benefit a structured process for business information systems analysis (see also Sect. 4.8.2).

Content Item “Business Goals & Restrictions”. In this content item, we formulate steering goals that have to be achieved by the execution of business processes, and further rules and constraints that have to be preserved during this execution, the latter resulting from the principles in the business vision.

In particular, the *Business Restrictions* includes not negotiable restrictions on the business processes, their structuring, and their interplay with information systems. The restrictions arise from the organisations’ principles and any kind of external influences, such as laws. In the item *Business Goals*, we define statements of intents at some future point in time [RR99]. Business goals are means to motivate the definition and decomposition of business processes and the inference of requirements in a value-oriented manner.

Content Item “Business Capabilities”. The business capabilities is an essential part of the business specification. Within the *Business Service Model*, we capture a logical representation of (repeatable) activities and define a black box description of the work that is carried out as a result of a set of business activities [Gro09b]. The *Business Process Model* includes a description of the the glass box view of business services by giving a description of the work flow (the business activities) performed in the provision of the business services. This content item serves as a means to understand in detail what business behaviour underlies single services and how this behaviour can be supported by information systems (see also Sect. 4.2.2.1).

Content Item “Business Information Model”. The business information model includes a description of the data that is processed and referenced during the execution of the business processes, respectively services. *Business Objects* include a

4.4 Artefact Type: Business Specification

description of unique objects of reality that can be of material (physical) nature, such as a “Course Attendee”, or of immaterial nature, such as an “Booking Request”. *Information Objects* include a description of the specialised information (data) content of a business object that is processed and that can be reproduced by an information system.

Content Item “Business Roles”. We describe all the roles that are directly or indirectly related to the business and that have a specific interest towards the project. We describe via *Stakeholders* a unique individual, group, or organisation that represents a customer-side or customer-related role. Stakeholders have an interest in the project and are able to mobilise resources to affect its outcome in some way [Wie03]. They are actively involved in a project having their interests expressed by business goals that, in turn, may be positively or negatively affected as a result of project execution or successful project completion [Cle05, Smi00, GBB⁺06].

With *Process Owners*, we describe individuals that have the responsibility for the performance of chosen business processes in relation to a set of business goals. They have the authority on the business processes and their execution and the ability to make necessary definitions and changes.

Finally, we describe via *User Groups* groups of individuals with a specific proficiency directly performing a set of business tasks and potentially interacting with information systems (see also [KCH⁺90]).

Content Item “Business Demands Analysis”. For each of the business services and involved processes, we describe within the business demands analysis

1. potential costs that are caused by the performance of activities and costs that arise from a system’s development in order to support this activity,
2. resulting business values that arises when achieving defined business goals, and
3. upcoming business risks that threaten the business.

These three indicators comprehensively justify any decisions during BISA activities in terms of decomposition and modification of the current business processes and finally the decision about if to automatise envisioned business processes. The business demands analysis addresses the question “can modifications of business processes and the support of the like by information systems achieve a business value the stakeholders are ready to invest for?”. Hence, it also serves the prioritisation of resulting requirements (see also section 4.5.2.1 introducing the requirements attributes).

Content Item “Business Glossary”. The glossary includes a description of (the semantics of) project-specific terms. These terms arise from the customer’s domain and its underlying business process logic. As terms that are familiar to contractors are often not familiar to customers (and vice-versa), and the semantics of terms within one industrial sectors might differ from the semantics of same and similar terms within another industrial sectors, this content item defines an individual, but consistent set of terms upon which the subsequent communication can be based on. This content item thus enables

1. an effective communication that is based on a well-defined vocabulary, and
2. the prevention of unnecessary misunderstandings that might expand the time schedule and the budget of a project.

4.4.2 Content Model

We now introduce the concept model of the business specification and structure the section according to the different modelling views introduced in Sect. 4.2.1.1. We begin with the concepts for structure modelling. Afterwards, we introduce the concepts for behaviour modelling, extend this view with concepts for data modelling, and conclude with a description of the business goals and the roles. The comprehensive concept model can be taken from appendix B.2.1.

4.4.2.1 Concepts for Modelling Structure

Figure 4.10 illustrates the excerpt of the concept model relevant for structure modelling. On top of the figure, we depict the concepts and relations used in the content item *Organisation Structure & Business Domains*. The bottom part of the figure illustrates further content items to which we have content dependencies.

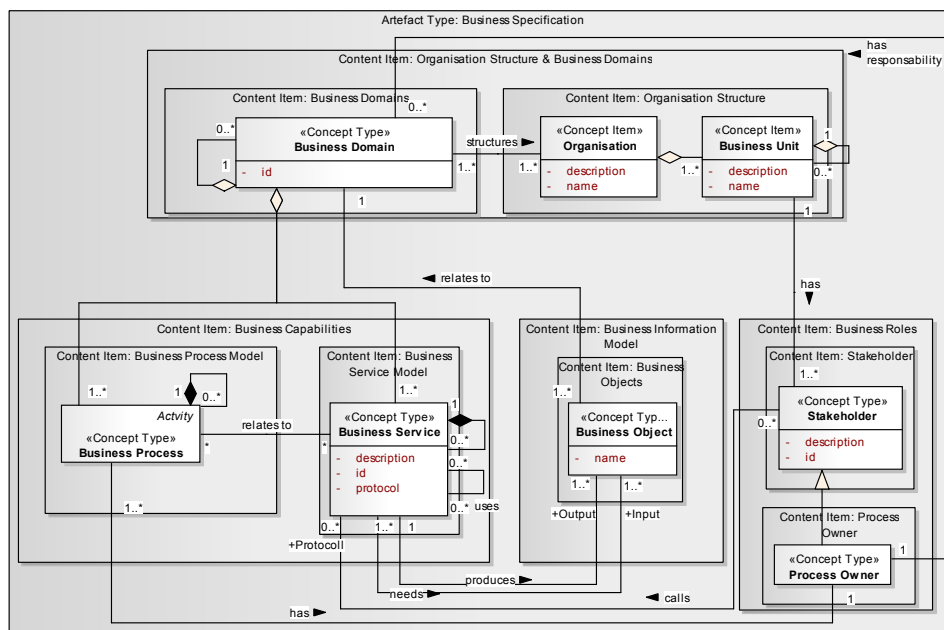


Figure 4.10: Excerpt of the concept model considering structure modelling in the *Business Specification*. The bottom part of the figure illustrates the dependencies to the concepts for behaviour modelling (see, e.g., Fig. 4.11).

As already mentioned, the organisation structure includes a description of the envisioned organisations and the (decomposed) business units being under management of at least one stakeholder. The business domains, however, take, in contrast to business units, a functional perspective clustering chosen business processes that underlie high cohesion. Hence, we explicitly capture not a relation to the stakeholders, but to process owners (being inherited from stakeholders, see also the next sections).

We use business domains as a means to logically group particular sets of business processes, regardless of the structure formulated via business units. As a consequence of further concepts relating to business process, business domains additionally group also these concepts. We refer, in particular, to business objects and to process owners that take the responsibility for the processes grouped by a busi-

4.4 Artefact Type: Business Specification

ness domain (and thereby for the business domain itself). From a methodological point of view, the business domains can be defined on the basis of these concepts. Finally, regarding the choice of the term, we intentionally use *business domain* instead of *domain* (e.g., as done in [EHH⁺08]) to avoid misinterpretations with similar terms like *application domain*. Regarding the concepts, there are two related approaches: *Problem Frames* [Jac00] and *Viewpoints* [KS96]. Problem frames offer formal means of structuring the environment that surrounds a system into so-called frames with a set of pre-defined patterns. Viewpoints are means of communication that structure requirements in relation to single stakeholders and their attitude towards the system. The least common denominator are the business processes that have to be supported by the system (structured into business domains). They represent the main interest of single stakeholders and, at the same time, they can be seen as a specific subset of problems to be solved. Hence, business domains can be compared to problem frames that structure business processes, and to view points, because they are related to a specific process owner.

4.4.2.2 Concepts for Modelling Business Behaviour

We now introduce the concepts for the modelling view *behaviour* within the business specification, i.e., the concepts used to create business process models and business service models.

There exists a variety of complex concept models for business processes covering the specialities that arise from available description techniques; for example, the *Business Process Definition Metamodel* [OMG06a] of the Object Management Group or the framework proposed by List et al. [LK06]. We use, however, a concept model as a means to abstract from such specialities and to keep the complexity to a minimum level (see also Sect. 2.5.1.1).

Hence, we introduce a concept model that does not define every detail given by available description techniques, while taking their least common denominator into account.

So far, we discussed in Sect. 4.2.2.1 the theoretical duality of services and processes as they are both similar notions to express same functional behaviour at different levels of abstraction. From a modelling point of view, however, we distinguish services and processes in their concepts as illustrated by Fig. 4.11.

The excerpt of the concept model shows the concepts of business processes and business services, by now without the associations to further concepts. These are introduced in subsequent sections and will complement the discussions in this section.

Business Services. A business service is a logical representation of the activities that are performed in order to provide the services, i.e., it represents the black box view on the work that is carried out. Business services define a business capability provided by the execution of one or more business processes satisfying at least one business goal (see Sect. 4.4.2.4).

A business service can be hierarchically decomposed and maps an input to an output [BKM07]. Besides the relation to business objects (Sect. 4.4.2.3) and the one to business processes, a business service relates to the concept of a stakeholder (Sect. 4.4.2.4) that represent a provider or a requester of a business service.

Finally, each business service includes, besides an identifier and a (service) description, a collaboration contract that optionally defines (selected) sequence of steps in which a service can be used by the requester. This contract is conceptually expressed by the attribute *Protocol*.

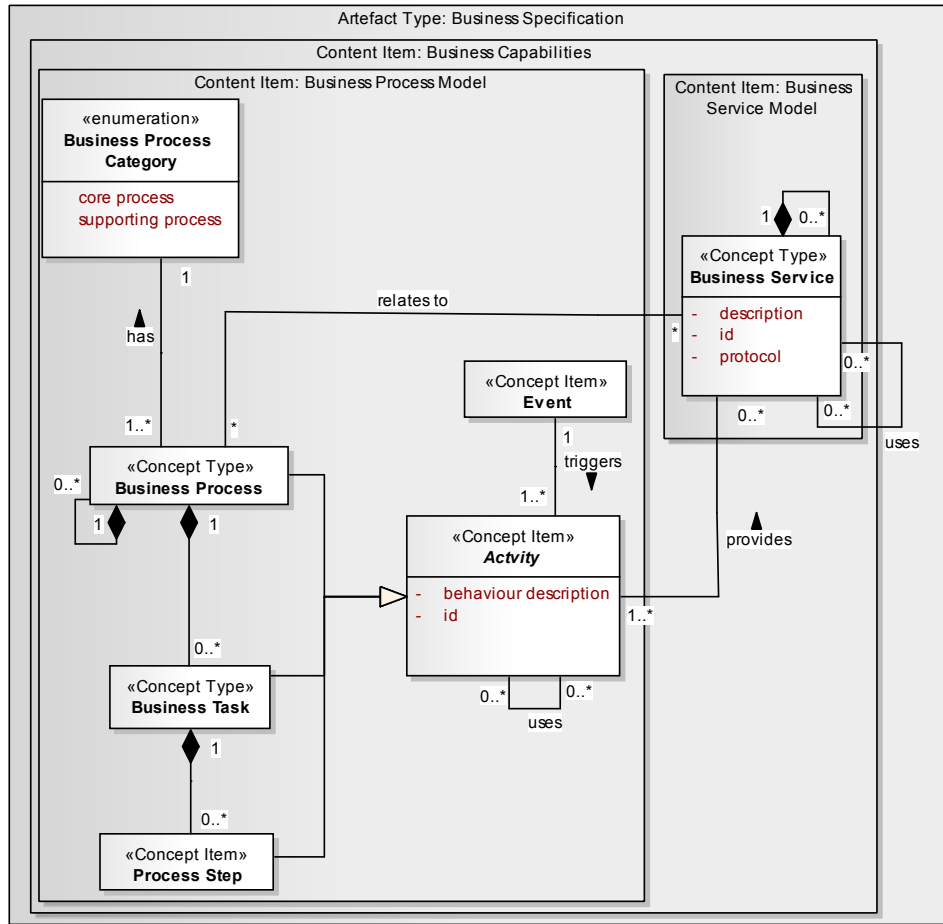


Figure 4.11: Excerpt of the concept model considering behaviour modelling in the *Business Specification*. Further associations are introduced in subsequent sections (see, e.g., Fig. 4.12).

Business Processes. We consider a business process model as a collection of all instances of the activities and their relations, performed by roles in order to produce some outcome of value. For the concept model, we take into account the work of Thurner [Thu04] that defines a *business process* by the following ingredients:

1. **Activities:** A business process consists of a set of activities that are executed and that define (partial) behaviour.
2. **Work flow:** The activities within a business process are triggered by an event and executed in a specific sequence. An activity can thereby *use* the next defining causality. Depending on the chosen description technique, the *uses* association can correspond to control nodes to express variations in the work flow (e.g. by joins or splits), as shown in [LK06].
3. **Processed data:** During the execution of the activities, a specific set of information is processed in terms of being created, read or modified.
4. **Participating roles:** Business processes are defined and administrated by specific roles and also performed by specific roles.
5. **Motivating business goals:** Each activity has to satisfy at least one business goal.

4.4 Artefact Type: Business Specification

The relations to the processed data is shown in detail in the following Sect. 4.4.2.3, the relations to the roles and to the goals in Sect. 4.4.2.4. As introduced in via the levels of abstraction, we distinguish a concept for representing the black box view, and two concepts for the glass box view:

- *Business processes* (black box) are administrated by process owners in order to achieve a business goal, while we decompose business processes to
- *business tasks* (glass box), that each provide a description of related atomic (not decomposable) *process steps* being performed, in turn, by single user groups to achieve process-related goals.

Compared to the concepts proposed by Cockburn in [Coc00], a business process corresponds with its encompassed tasks to a “black box business use case”, while the process steps represent the “white box business use case”. Example 4.5 illustrates the three main concepts used for modelling business processes.

The responsibility for the single atomic process steps rests within the (content item) business process model with the user groups, while these follow particular goals used as a stop criterion for decomposition. This also means that the work flow description provides detailed steps to be performed, but without involving systems. Such a description of how parts of the work flow (and consequently the business service) can be supported by a system is provided in the requirements specification (see the system vision in Sect. 4.5).

In particular, we refer to the use case model and the information system service model, that both define the (black box) behaviour of a system. In order to infer such a system behaviour description, we conceptually *realise* the single process steps by either a system action or by an actor action. Both concepts are means to specify which behaviour remains to users and which behaviour is reproduced by a system. Appendix B.2.3 illustrates in detail these dependencies between the concepts at hand and the ones of the requirements specification.

In summary, we see a business process according to [Thu04, Coc00, EHH⁺08] as a cross-functional collection of all possible instances of related activities, performed across multiple business units, and contributing to at least one business goal. It is administrated by a process owner and serves to directly or indirectly produce a product or to provide a business service.

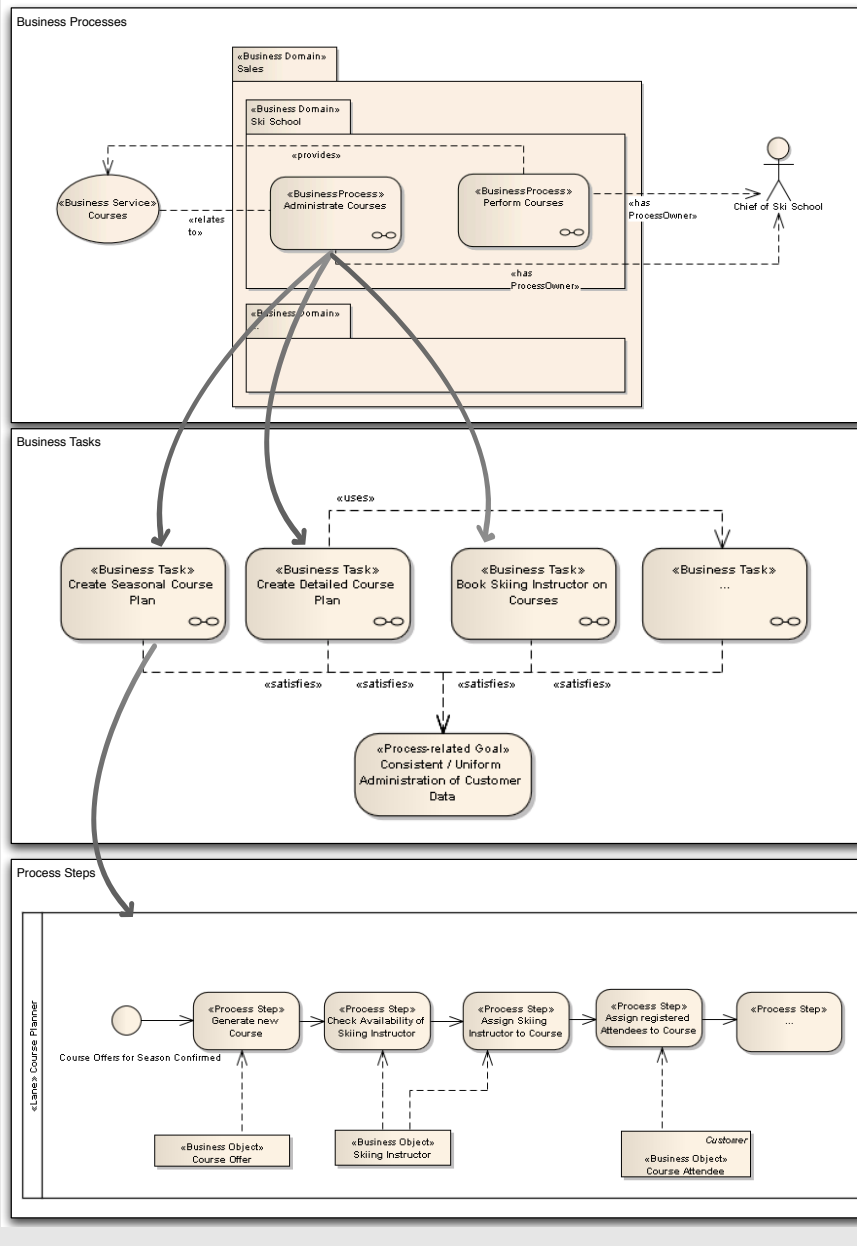
The activities are triggered by an event³ and may use other activities to transform (produce, read, modify) one or more business objects and information objects. Business processes can be hierarchically decomposed until reaching the granularity of business tasks.

Business tasks are partial units that are encompassed by a business process. Each business task consists of an ordered set of related, atomic, not decomposable process steps, while these are

- performed by exactly one user group that aims at achieving at least one process-related goal, and that
- can be supported by an information system.

Finally, as depicted by the concept model in figure 4.11, we distinguish furthermore two categories of business processes: *core processes* and *supporting processes*. The business process category defines if a business process directly or indirectly produces a product or provides a business service. The first is declared as a core process, the latter as a supporting process (see also [EHH⁺08, WGWP02, LK06]). Core processes have an operational character and directly provide at least one business service. Supporting processes are indirectly involved in terms of making a weak, but necessary contribution to achieving business goals and providing related services.

³ A change of state or a temporal event.

Example 4.5. Exemplary Business Processes at different Levels of Abstraction

4.4.2.3 Concepts for Modelling Data

We now extend the concepts introduced in the foregoing section with concepts for data modelling within the business specification. Figure 4.12 illustrates on the right side two concepts and their relation with the previously introduced business capabilities.

The content item at hand captures an explicit representation of the entities (data object), which are produced or modified during the execution of business activities and, thus, which are referenced within the corresponding business process or business service models (comparable to a data dictionary). As a consequence of

4.4 Artefact Type: Business Specification

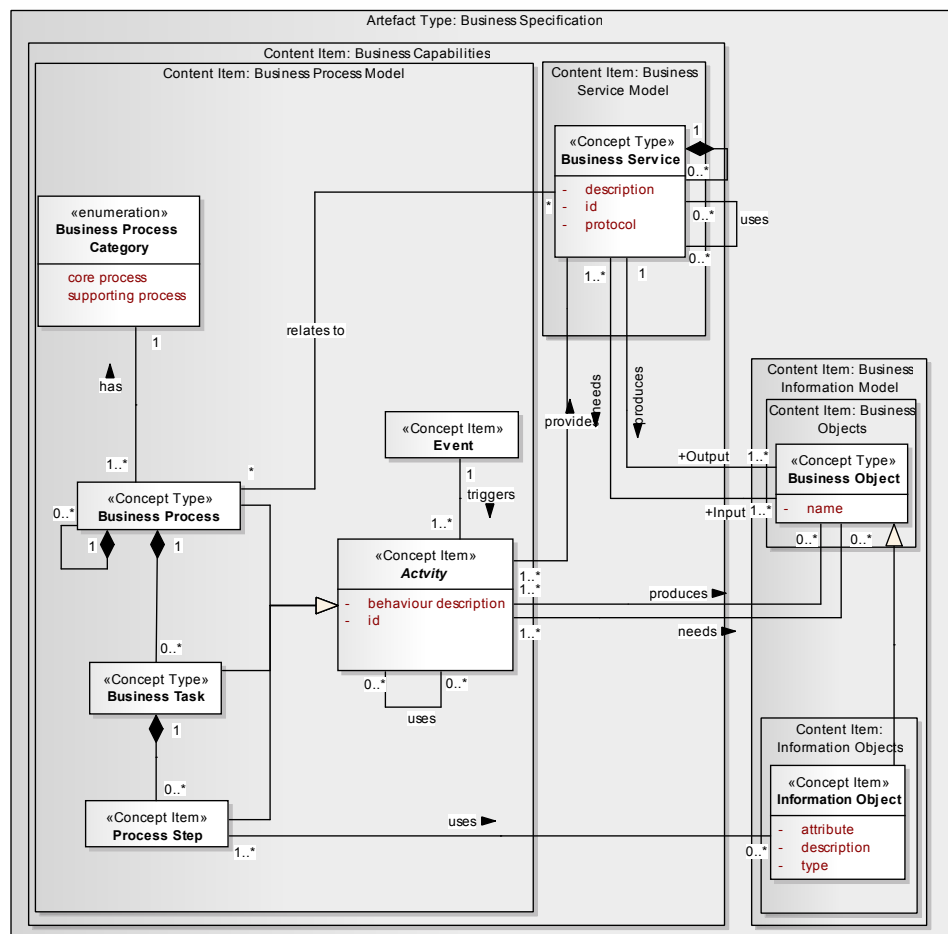


Figure 4.12: Excerpt of the concept model considering data modelling in the *Business Specification*. The left part of the figure illustrates the dependencies to the concepts for behaviour modelling (see Fig. 4.11).

the introduced concepts for modelling business behaviour among different levels of abstraction (see the section before), we distinguish two concepts for modelling data within the business specification: *business objects* and *information objects*.

Habitually spoken, a business object is a representation of an object of reality, i.e., the type of information that is processed within business process and / or business service models. An information object is the content representation of the business objects including, in turn, a set of attributes.

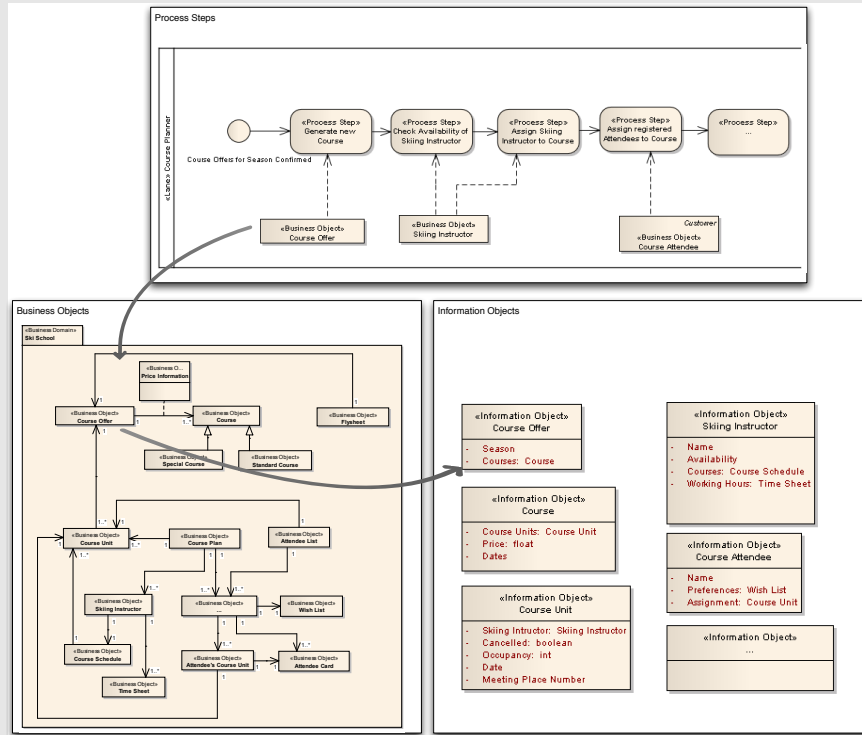
A business object thus represents the (data) type of the information being processed within, e.g., a work flow, where typed messages are processed and exchanged. Its definition results from the naming convention chosen within the corresponding activity in which the object is created (respectively the substantive in the activity, such as “create *Order*”). The information object, in turn, is conceptualised as a specialisation of a business object that includes the content information of the business object via a set of attributes.

Since both concepts relate to the concepts for modelling behaviour with, e.g., work flows, we do not define in the concept model an explicit relation between several business (information) objects themselves. These indirectly arise as a consequence of the actual work flow, i.e., the particular sequence of single process steps to which

the objects are related (see also example 4.5 on page 119).

Example 4.6 illustrates an excerpt of the business objects and the information objects, which we use in the process steps of the foregoing example 4.5.

Example 4.6. Exemplary Business and Information Objects



Regarding the relation to the business services, the concept of business objects is of a particular interest. The reason is that services are characterised by their mapping of (typed) inputs on outputs. For instance, via the interface descriptions, respectively port descriptions. In that case, the naming conventions in the business objects result from the actually defined interfaces (ports) of the services.

Therefore, not every business object needs from a methodological point of view a specialisation by means of information objects. These are of particular interest when explicitly modelling the work flow with, for example, interaction patterns as done by Hummel et al. in [HT09], where typed messages with content information (attributes) are exchanged.

Finally, regarding the transition between the business (information) objects to the models that specify the behaviour of an information system (found in the requirements specification), the explicit definition of information objects is necessary in order to enable their reproduction as part of a detailed data model (named in our context *Information System Objects*).

4.4.2.4 Concepts for Modelling Business Goals, Restrictions, and Roles

So far, we discussed concepts considering the modelling views *structure*, *behaviour*, and *data*. We now describe the concepts used for modelling the environment by means of business restrictions, business goals, and business roles. Figure 4.13 illustrates on the left side the concepts for modelling business restrictions and goals.

4.4 Artefact Type: Business Specification

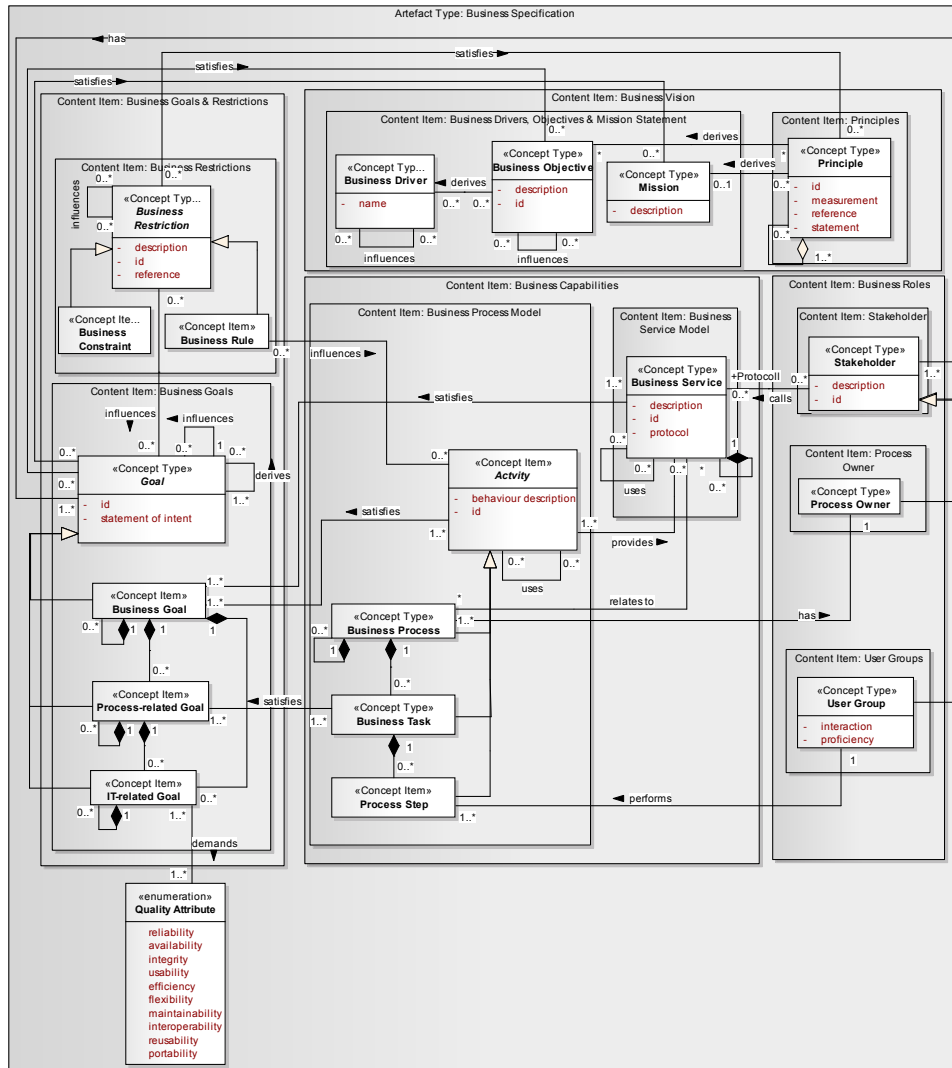


Figure 4.13: Excerpt of the concept model considering modelling of business goals (left part of figure) and business roles (right part of figure) in the *Business Specification*. The middle part of the figure illustrates the dependencies to the concepts for structure modelling (see Fig. 4.10) and behaviour modelling (see Fig. 4.11).

On the right side of the figure, we illustrate the concepts for modelling roles. In the centre of the figure, we illustrate the associated concepts for modelling structure and behaviour, but omit for reasons of complexity the previously introduced concepts for modelling data (that have no relations to the concepts discussed in the section at hand).

Business Roles

We already introduced in Sect. 4.4.1 the three concepts used to specify the roles. We take with *Stakeholders* the most general view on roles. Stakeholders represent individuals, groups, or organisations with a specific interest in the project that is expressed by goals [Cle05, Smi00, GBB⁺06]. We conceptualise thereby stakeholders by their relation to goals and in addition by their relation to business services

that they potentially call.

We specialise the concept of stakeholders with two further concepts. First, with *Process Owners* that represent individuals having the administrative sovereignty over business processes and included business tasks. Second, with *User Groups* that represent individuals participating in the execution of the business tasks. User groups abstract from concepts used in description techniques to represent responsibilities for the performance of single process steps, such as done via “Lanes” in BPMN [Bus03]. These user groups are then *realised* by actors within the requirements specification to describe how the user groups (and external systems) intent to interact with the information system under consideration (see also appendix B.2.3 and the requirements specification in Sect. 4.5).

Business Goals & Restrictions

Business restrictions define rules and constraints that restrict the execution of business activities while goals define statements of intent to be achieved by the execution of business activities (see also Fig. 4.13). Both concepts are introduced in the following.

Business Restrictions. Business restrictions define any kind of external influences that affect the organisation, potentially relating to chosen principles; for instance, an influence that arises from a law to be preserved as a consequence of the incised market while this rationale (the law) is conceptualised via a *reference*. We specialise in dependency to the concept’s concern business restrictions to *business rules* and *business constraints*.

Business constraints represent not negotiable limitations for any kind of solution regarding business process logic and information systems. The constraints state limitations placed on a solution, or an aspect of the current state that cannot be changed by the deployment of the new solution. Hence, the concept can (semantically) influence stated goals. Business rules are actionable directives that define or constrain behavioural aspects of the business and that directly support a goal. A business rule is intended to assert business structure or to control or enable the activities of the business [HH00, IIB09], expressed by the association to the activity concept. A business rule describes standard procedures in the business processes having the nature of “action enablers” triggered by a certain condition (comparable to “if then else” - statements, see [Wie03]).

Goals. Goals are, in general, seen as a rationale for business activities and system properties resulting in an expected value [RR99]. We use the concept to specify prescriptive statements of intent at some future point in time in relation to particular stakeholders. For the concept model, we take into account the goal modelling approach *Knowledge Acquisition in Automated Specification of Software* (KAOS) [vL03, vL04, vL09]. This approach introduces the basic concepts of goals by means of decomposition and various types of influencing relationships. A goal potentially influences another goal and can be derived from another goal. In addition, we take into account domain-specific approaches, like *Quasar Enterprise* [EHH⁺08]. These approaches define hierarchies of goals while separating different types of goals according to their concern and (influence) relation to further concepts within corresponding architecture framework (like customer-related goals or process-related goals).

By considering our refinement hierarchy introduced in Sect. 4.2.2, we distinguish during the decomposition of goals three particular types. First *business goals*, which

4.4 Artefact Type: Business Specification

define a rationale for general business capabilities including also financial and customer-related intents. Second, *process-related goals*, which particularly relate to business tasks and included work flow descriptions. Third, *IT-related goals*, which directly demand high-level system (quality) characteristics in order to support the process-related goals (see also Sect. 4.2.2.1 where we give examples).

Regarding IT-related goals, we define the relation to one or more *Quality Attributes*, as defined by the ISO Std. 9126 [ISO03] and the ISO Std. 25030 regarding (product) quality requirements [Boe08]. A quality attribute reflects a composition of perceptible characteristics, e.g., “Efficiency”, exhibited by a system and its environment due to its properties. Hence, IT-related goals are in the primary scope of system quality requirements that often can be directly derived from such goals (see also the requirements specification in Sect. 4.5).

4.4.3 Syntax

A description technique can be taken for the representation of the content items, if it offers means to express the concepts and relations defined by the concept model in Sect. 4.4.2. Different description techniques can vary, however, in their emphasis of different concepts and relations of a particular content item (see also the discussion in Sect. 3.3.2.1).

Table 4.2 on page 125 thereby summarises, where possible, sets of representative description techniques for each of the introduced content item. Where reasonable, we describe criteria for variations in the choice of description techniques; for instance, goals can be specified textually or as part of graphs while the administration of the graphs should be justified by a certain complexity.

Most of the content items, however, are informally represented by means of natural language. This mainly arises from the target groups and affects those content items that are of interest in early development stages (like the business vision). A variation of an informal and a formal specification of a content item is exemplarily given by the business restrictions, where logic-based formalism can be applied for representing business rules.

Regarding the business process model, we also summarise description techniques with different degrees of formalisation, while a detailed discussion on these techniques can be taken from Thurner [Thu04] and from our contribution [MFK09].

In addition, we describe variations in the choice of description techniques by differentiating whether the techniques put emphasis on concepts for explicitly modelling control flows, data flows, or whether no differentiation is made (where concepts for both are given same attention).

Although our concept model may imply with the notion of work flows a certain emphasis on concepts for modelling control flows, there exist project-specific situations that argue for the explicit definition of data flows. For instance, if the organisation (respectively business domains) is structured according to business objects and the envisioned information system will emphasise the processed data rather than the enabled work flow and its specification via interaction scenarios.

Table 4.2: Representation of content items of the business specification.

Content Item and Concepts	Variation Criteria	Description Techniques
B. Drivers, Objectives & Mission St.	None	Informal graphs or textually within list
Principles	None	Hierarchically organised taxonomy
Expert Problem Description	None	Informally in natural language
Scope & Limitations	None	Informally in natural language
Enterprise Overview	None	Informally in natural language
Business Service Overview	None	Graph emphasising the relationships in-between the single business services or informally in natural language, structured, e.g., by a table
Organisation Structure	None	Organisational Diagram
Business Domains	No tool support Tool support	Informally in natural language Package structure
Business Restrictions	None	Informally in natural language. Business rules optionally via predicate logic, in some cases via temporal logic due to actionable nature (depending on target groups).
Business Goals	Low complexity High complexity	Informally in natural language, structured, e.g., by tables Goal graphs (e.g. via KAOS [vL03, vL04, vL09])
Business Service Model	None	Event/Response Tables [Wie03], Input/Output Tables [HT09]
Business Process Model	Data flow orientation	Business Process Net [Thu04], Data Flow Diagram (Structured Analysis) [DeM79], Message Sequence Chart (MSC) [IT99], UML Sequence Diagram [OMG10a, OMG10b]
	Control flow orientation	Event-driven process chain / EPC (ARIS) [Sch91, Sch02, SJ96], FUNSOFT net [DG96], Business Process Model and Notation (BPMN) [Bus03]
	Data and control flow	ProBaMa [LSS10], UML Activity Diagrams [OMG10a, OMG10b]
Business Information Objects	Low complexity High complexity & tool support	Informally in natural language (via tables) UML class diagram [OMG10a, OMG10b]
Business Roles	None	Informally in natural language, the overview can be expressed, e.g., via an onion model (further techniques are summarised in [Rup07])
Business Demands Analysis	None	Informally in natural language via table, in which each entry documents demands of single business services (see also [MFK09])
Business Glossary	None	Informally in natural language via table

4.5 Artefact Type: Requirements Specification

We described in the foregoing section the business specification by introducing the structure model, the content model, and exemplary ways for representing the artefact type. In this section, we describe the requirements specification following the same structure. Similar as done within the business specification, we stick in this section to the definition of those concepts types that have been introduced in Sect. 4.2.2, while detailed discussions can be taken from our previous contribution in [MFK09].

4.5.1 Structure Model

Figure 4.14 illustrates the structure model of the requirements specification. Same as it is the case for the structure model of the business specification, we give with the figure at hand the most abstract view that can be taken on project-specific exemplars of the content items. Each item has a 1:1-dependency. The different possibilities for structuring the content by means of additional, multiple content items (reflecting project-specific contents) are introduced in appendix B.1.2.

In the following, we introduce the single content items of the requirements specification.

Content Item “System Vision”. While the business specification comprehends the description of a business environment to be realised (with processes, restrictions, ...), the requirements specification describes how an information system shall fit into this environment. The system vision serves to specify the alignment between both the business and particular systems.

The vision defines how one information system will support certain business processes and preserve stated restrictions and goals. From a methodological point of view, the system vision defines an important agreement whose creation and acceptance is coupled to one particular milestone. The system vision serves as a means for scoping and the detection of moving targets. For this purpose, the system vision consists of two major content items:

- *Summary of Business Specification* summarising selected contents of the business specification necessary to define the (functional) scope of one particular system (please refer also to Sect. 4.4).
- *Scope of the Information System under Consideration* describing high-level functionality of this system in relation with the selected items of the business specification.

The latter prescribes a system under consideration in its breadth by specifying the system’s boundary and listing its major use cases as well as its information system services. This initial overview is specified in full in the subsequent content item *Information System Requirements*, e.g., by defining scenarios of interaction for each of the listed use cases.

To define the scope of the information system, we consider five content items:

1. *System Overview* that specifies the system’s boundary by defining the interdependencies between the system and its environment, given by the actors (realising user groups and external systems). These interdependencies define what actors intend to interact with the system under consideration and how they intend to do it (via indicated directions of communication flows).
2. *External Systems* that describes systems including one or more applications that directly or indirectly interact with the envisioned information system in order to realise the work flow of a business process.

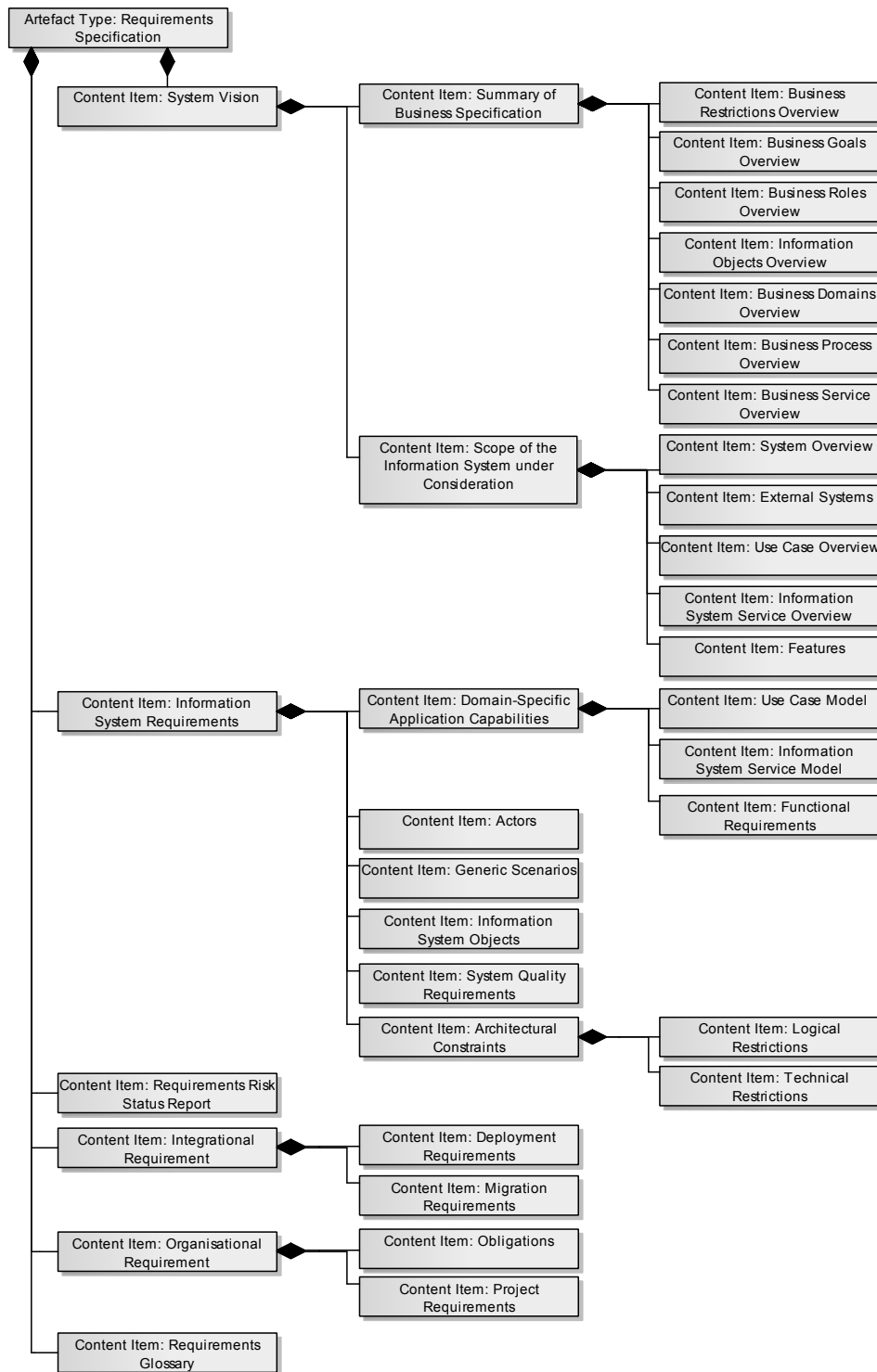


Figure 4.14: Structure model for the artefact type *Requirements Specification*.

3. *Use Case Overview* that provides an overview of the use cases that directly can be performed by actors. It illustrates what business tasks will, at least partially, be automatised by the information system and defines the relation-

4.5 Artefact Type: Requirements Specification

- ships between actors and use cases, as well as in-between the use cases.
4. *Information System Service Overview* that summarises all information system services that are offered by the envisioned system and that result from the necessity of supporting the business processes, respectively business services.
 5. *Features* that summarise prominent or distinctive user-recognisable aspects, quality, or characteristics of an information system that are related to a specific set of requirements, whose realisation enable the features [CHS08, KCH⁺90].

Please note that while the first four content items encompass interdependencies of a system with its environment and summaries of self-contained concepts for modelling functional behaviour (like summarised use cases), features take a special role. In contrast to the area of product lines development, in which features are hierarchically organised by same or similar interacting product characteristics [KCH⁺90] illustrating possible variations of the like, we see in our envisioned domain features as single recognisable characteristics of a system grouping existing requirements. Each characteristic defines one particular view onto a set of existing requirements that are specified with further self-contained concepts; for instance, a feature “Transaction Safety” grouping a set of architectural constraints. Hence, a feature groups a recognisable set of logically related requirements under one specific characteristic [Wie03]. It thus relates to further concepts rather than being a self-contained one that abstracts from a concrete description technique used to express proper concerns.

Content Item “Information System Requirements”. With information system requirements, we consider all concepts used to express demands on system’s functionality, properties, conditions, architectural constraints, and the system’s environment. Most of a requirements specification’s content proposed by the (domain-independent) IEEE Std. 830-1998 [IEE98] is incorporated by this content item.

The *Domain-Specific Application Capabilities* incorporate a description of the system’s functionality that is necessary to realise a (seamless) functional support of the business process logic. Regarding the project-specific exemplars of the artefact type, this item can be reproduced for each of the business domains (see also appendix B.1.2).

We conceptualise system’s functionality with three concepts and, thus, three content items: A *Use Case Model* to describe the interaction scenarios between actors and the system for each of the use cases stated within the system vision; *Functional Requirements* to describe single system’s actions expected as a response to specific events [Dav93]; *Information System Services* to describe functionality via single services as a complementary means to interaction scenarios.

The *Information System Objects* comprehend a description of the system-side reproduced information objects (see Sect. 4.4.2.3). They are processed by the system when calling an information system service and / or interacting with the system, described as part of a use case.

The introduced content items are functionally motivated and relate to business processes to be supported, whereby the content item *Generic Scenarios* extends this view to express quality-related usage scenarios. This content item relates to *System Quality Requirements* that constrain a system and its (development and usage) environment in order to support chosen activities. *Architectural Constraints* include a description of (structural) architectural restrictions, affecting the logical and / or the technical architecture and, thus, separated by the two content items, depicted in Fig. 4.14.

Finally, we represent via *Actors* all the external entities that interact with the future system, realising the user groups and the external systems.

Content Item “Requirements Risk Status Report”. The requirements risk status report includes a description of all risks that are related to project-specific requirements. The report serves risk management to monitor risks and to define appropriate counter measures already during early development activities. We rely on the work of Islam [Isl09, IHMFJ09] that considers the conceptualisation of requirements risks on the basis of the artefact model at hand [Isl11, BFI⁺09].

For each of the requirements risks, he proposes to capture

- the risk itself including the related factors as a root cause for the risks,
- the risk score to determine the criticality of risks while also referring to the risk impact that represents the severity of the risk, and
- the risk mitigation points to capture appropriate treatment actions to control the risk.

Content Item “Integrational Requirements”. This content item includes restrictions on the delivery and integration of the system w.r.t. the process and technical details serving the planning of release strategies. For this, the item consists of two self-contained items: *Deployment Requirements* describing demands towards the deployment procedure, constraining the process design of the deployment, and the technical environment during initial launch of the system or specific parts of it; *Migration Requirements* describing restrictions on the replacement or modification of one or more legacy systems. Migration requirements consider two aspects:

1. The displacement of legacy systems emphasising needs towards the procedure itself (e.g., considering sequential steps for displacing legacy systems).
2. The data migration from legacy systems to the envisioned one, covering the data elements to be transferred and used mapping rules for the information system objects.

Content Item “Organisational Requirements”. Organisational requirements include restrictions towards the overall project execution and its process. We consider two content items:

- *Obligations* including agreements, propositions, and exclusions between the parties within a development project. They encompass demands towards a specific action that has to be taken by a party.
- *Project Requirements* constraining the content and / or structure of selected artefact types and the process model, i.e., the definition of the milestones regarding time schedules, used infrastructure like mandatory tools, and compliance to selected standards and approaches like to the V-Modell XT.

Content Item “Requirements Glossary”. The glossary extends the one of the business specification with requirements-specific (mostly technical) terms, such as terms arising from architectural constraints.

4.5.2 Content Model

In the following section, we first introduce the concept types to which we refer for the domain-specific interpretation of the term *requirement*. Afterwards, we introduce the concept model according to the different modelling views, extending

4.5 Artefact Type: Requirements Specification

and specialising the introduced requirements concepts. We put emphasis on the concepts for modelling information system requirements while the comprehensive concept model can be taken from appendix B.2.2. A description of the concepts that subsequently are out of scope (from a conceptual point of view mainly trivial ones) can be taken from the structure model. The dependencies to the concepts of the business specification are introduced as part of the different modelling views, why we also omit the explicit introduction of the system vision that considers these dependencies. A comprehensive view on the dependencies between requirements concepts and ones of the business specification can be taken from appendix B.2.3. Finally, note that we omit, in contrast to the business specification, the explicit introduction of a structural view via business domains, since these affect the concepts of the requirements specification similarly as it the case for the business specification (structuring behaviour).

4.5.2.1 Basic Requirements Concept Types and Attributes

Figure 4.15 illustrates the basic requirements concept types and attributes.

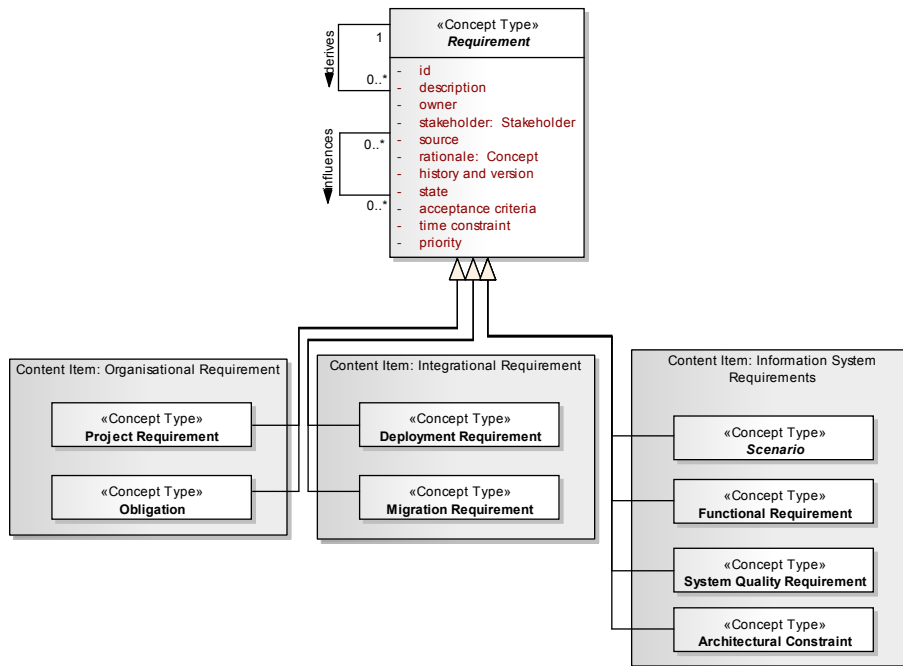


Figure 4.15: Basic requirements concept types and attributes.

We conceptually consider only those concepts as requirements which are used to describe properties of information systems, their integration, and their development process. Thus, we consider three different content items. We define *Organisational Requirements* and *Integrational Requirements*, that both affect management and planning activities, and *Information System Requirements* that consider properties of the system. For the latter, we make use of *Scenarios*, *Functional Requirements*, *System Quality Requirements*, and *Architectural Constraints*.

In particular, we conceptualise scenarios as a means to describe any kind of interactions between actors and the system and to express via the system's intended use its expected external behaviour. This concept is then specialised to further ones: to structured scenarios illustrating interactions grouped by functionally motivated

use cases, and to generic scenarios. Although (information system) services are habitually understood as the formalisation of user requirements [Rit08], this concept doesn't directly inherit from the requirements concept type. Instead, we conceptualise services as a complementary means to model functionality in relation to scenarios.

Requirements Attributes. Table 4.3 illustrates the requirements attributes to which we refer in the concept model.

Table 4.3: Basic set of requirements attributes.

Attribute	Description
ID	Identifier of a requirement.
Description	Assertion of a requirement used to transfer a requirement into corresponding classes (respecting the concept model).
Owner	Contractor-side individual being responsible for decision taking (applying the requirement or of denying it).
Stakeholder	Customer-side individual, group, or institution having the responsibility for the requirement.
Rationale	Justification of requirements lying in the satisfaction of further concepts, such as business processes or goals.
Source	In cases in which requirements cannot be rationalised by a specific concept, (informal) sources of elicitation include workshop protocols, emails, or protocols of phone calls.
History and Version	The history captures a chronological view onto the performed changes (of versions) benefiting an understanding of the requirements' life cycles and stability.
State	Current state of a requirement during its life cycle. An exemplary set of states and transitions is as follows: <div data-bbox="630 1220 1268 1377" data-label="Diagram"> <pre> graph LR Proposed((Proposed)) --> Validated((Validated)) Validated -.-> Proposed Validated --> Assumption((Assumption)) Validated --> Denied((Denied)) Validated --> Accepted((Accepted)) Accepted --> Other([Other]) Other --> Applied((Applied)) Applied --> Released((Released)) InfoSysReq([Information System Requirements]) --> Designed((Designed)) Designed --> Implemented((Implemented)) Implemented --> Tested((Tested)) Tested --> Released NewReq((New Req)) -.-> Proposed </pre> </div>
Acceptance Criteria	One or more circumstances under which a customer accepts a solution for a requirement including a measurement (possible solutions and / or test cases) to objectively decide on the fulfilment of customers' expectations (comparable to the <i>fit criterion</i> in the VOLERE Requirements Shell [RR07]).
Time Constraint	Point in time or planned release when the requirement should be realised (important within time-boxing).
Priority	Importance of a requirement.

Note that these attributes are based on experiences made during the development of the approach (see Sect. 3.3.3) and on available approaches, like [RR99, RR07, Rup07]. Thus, there exist possibilities for variations in the attributes and especially for their representation. For representing the attributes, a trade-off is needed

1. between the desired precision of the attributes and the effort needed to create them (e.g., regarding the requirements priority that can be calculated in different ways), as well as between
2. the areas of validity of attributes and the effort needed to manage them (e.g., regarding the requirements priority being calculated for single functional requirements or for all requirements relating to a certain use case).

4.5 Artefact Type: Requirements Specification

Finally, in contrast to approaches like REM or the IEEE Std. 830-1998 (see Sect. 3.1.3), we capture assumptions as a particular requirements state rather than as an own content item in order to avoid redundancies in the created artefacts.

4.5.2.2 Concepts for Modelling System Behaviour

Figure 4.16 illustrates the concepts used for modelling system behaviour.

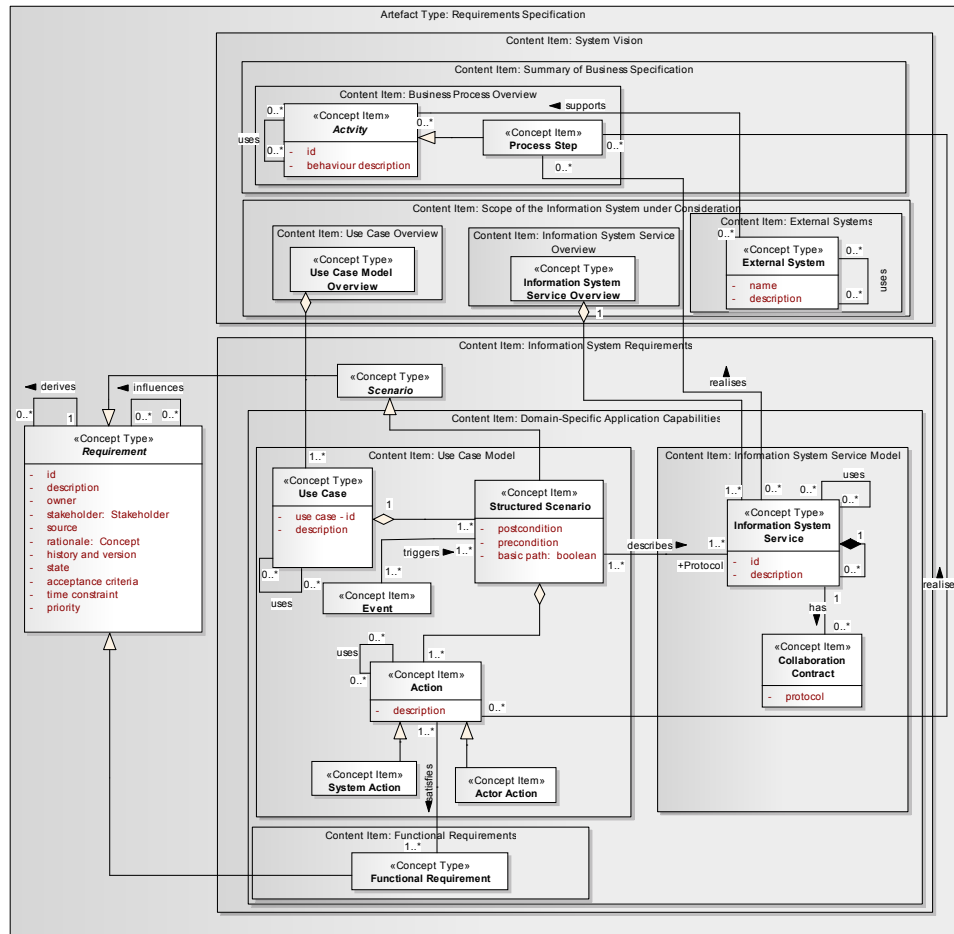


Figure 4.16: Excerpt of the concept model considering modelling of (system) behaviour in the *Requirements Specification*. The upper part illustrates the dependencies to the concepts of the business specification, captured within the content item *System Vision* (see also Fig. 4.11).

The upper part of the figure illustrates an excerpt of the system vision that relates to the domain-specific application capabilities on the bottom part of the figure.

We now describe the concepts used for the content items *Use Case Model*, *Functional Requirements*, and *Information System Services*. The first two concept types inherit from the requirements concept, the latter relates, in turn, relates to the use case model.

Use Case Model. The use case overview in the system vision groups a set of logically related use cases that can be performed by one or more actors, each showing

scenarios of system-supported interaction between actors and the system in context of the business processes. In the following, we introduce the concepts of the actual use case model. A summarising definition can be found in the glossary, appendix A.

According to Cockburn [Coc00] and Wiegers [Wie03], a use case is a discrete, stand-alone, and system-supported activity that is triggered by an event and performed by an actor in interaction with the system. It describes the observable behaviour and interaction of a system in response to an actor action.

A use case model serves as a means to understand the expectations of particular user groups towards the system in relation to the process steps, which they have to perform. Each use case in a use case model encompasses a collection of related scenarios, which we define as *structured scenarios* to differentiate them from the *generic scenarios* introduced in Sect. 4.5.2.5.

One (structured) scenario is a specific instance of a use case specifying an ordered sequence of actions that occur under certain conditions. Each action is either performed by an actor (*actor action*) or by the system (*system action*) while both *realise* a process step in the business process model. The actions within each scenario are single units of behaviour within a scenario, realising each an atomic process step and being themselves not further decomposable (in the requirements specification).

Example 4.7 on page 134 illustrates the realisation dependencies between a use case and the business tasks to be supported, and the dependencies between the actions within a structured scenario of a use case and the process steps of the corresponding business task.

The transition from the business process model to the specification of external system behaviour is, methodologically, performed by selecting which of the process steps shall be reproduced by a system action and which remain to the responsibility of the user groups, realised by actors (see also Sect. 4.2.2.1).

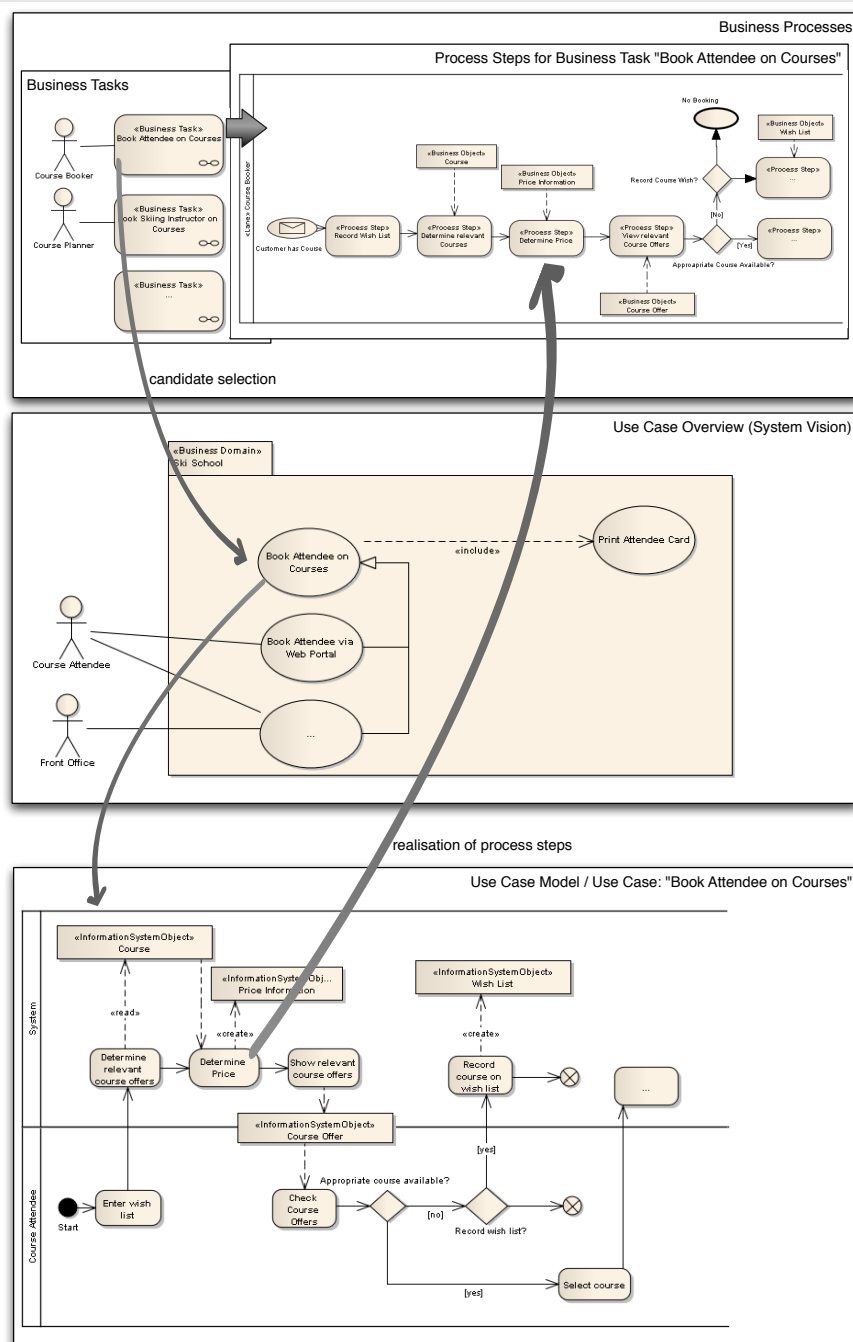
The conditions, which are related to the actions, describe the state of the system before the scenario is executed (precondition) and the state after the execution (post-condition). In addition, we distinguish in the concept model whether the scenario describes the basic (expected) path of actions or whether it is an alternative path.

In contrast to available modelling languages, like the UML [OMG10b], we omit in the concept model two concepts for the specification of a use case: control nodes and decomposition of actions⁴. Control nodes are reduced, similarly as done within the business process model, to an extendable *uses*-relation within the actions. We omit the decomposition of actions, since the actions already realise atomic process steps either by actor actions or by system actions. The decomposition of actions would thereby affect, if at all, system actions. However, since we see at this level of abstraction the system as a black box without the necessity of further refining system behaviour and relating system actions to single components, we omit this association. If we would consider in the thesis the system specification, respectively the specification of a logical component architecture, the decomposition could be expressed by a separate concept of a system use case, as introduced by Cockburn [Coc00], respectively by functions and function hierarchies (see also appendix B.2.4).

⁴ The decomposition is expressed in the UML superstructure [OMG10b] with concepts like *structured activities* and (atomic) *actions*.

4.5 Artefact Type: Requirements Specification

Example 4.7. Exemplary Use Case in Relation to Business Processes



Functional Requirements. Habitually spoken, a functional requirement is referred to as a single “the system shall do something”-statement. According to Davis, a functional requirement specifies the demanded external system behaviour given by one stimulus, followed by one expected response, as part of a reaction to the stimulus, independent from the underlying realisation of this response [Dav93]. A functional requirement thus corresponds to one particular actor

action, followed by a system action (process the same information system objects). As not all functional requirements are covered, from a methodological point of view, by use cases, we explicitly define the concept in addition to the one of actions.

Information System Service Model. The information system service model specifies each of the services summarised in the system vision. As discussed in Sect. 4.2.2.1 as part of the levels of abstraction, we use information system services as an explicit concept to define logical representations of single pieces of functionality of an information system [Gro09b]. A service can be called by an actor without a mandatory description of the relation to actors and of concrete interactions sequences. The interaction sequences can be optionally defined by the use of *Collaboration Contracts*, which define an ordered sequence to use the provided services via a protocol (see also Sect. 4.5.2.5). According to [BKM07, Bro05, Bro03, HT09, EHH⁺08], we make use of information system services as a concept that is complementary to the one of a use case. A service gives an abstract definition of single pieces of system's functionality supporting the realisation of process steps and / or business services. A single service can be hierarchically decomposed to the level of actions (also known as "elementary services" [EHH⁺08]) while mapping an input (stimulus) to an output (response). Please note, that although the behaviour description of an information system implies a set of states, we omit the mandatory definition of these states and thereby their conceptualisation, because of the envisioned level of abstraction. See also Sect. 3.3.3 for further information on the development of the concept model.

4.5.2.3 Concepts for Modelling Data

Figure 4.17 illustrates the concept type of an *Information System Object* and its relation to the previously defined concepts used for defining domain-specific application capabilities.

The upper part of the figure illustrates the relation to the concept of an information object (captured in the summary of the business specification).

An information system object realises information objects. Information system objects thus are a system-side reproduction of selected information objects that are processed (created, read, updated) when calling an information system service and / or executing a scenario in a use case (see example 4.7 on page 134). This relation is also reflected in the input / output - associations to the actions within the use case model and to the information system services. We refer with the attribute *type* to the information object from which the information system object is inferred.

In contrast to the information objects in the business specification (see Sect. 4.4.2.3), we explicitly define, as found in data models, the relation between single information system objects by means of (directed) relations with cardinalities between the objects.

4.5.2.4 Concepts for Modelling Actors

Figure 4.18 extends the behaviour view with the concept used to describe actors (on the right side of the figure).

An *Actor* conceptualises a unique role outside the system under consideration that is able to execute an action of a scenario and / or call an information system service. We distinguish between primary actors that provide and use information and secondary actors that only use information from the system as an input.

4.5 Artefact Type: Requirements Specification

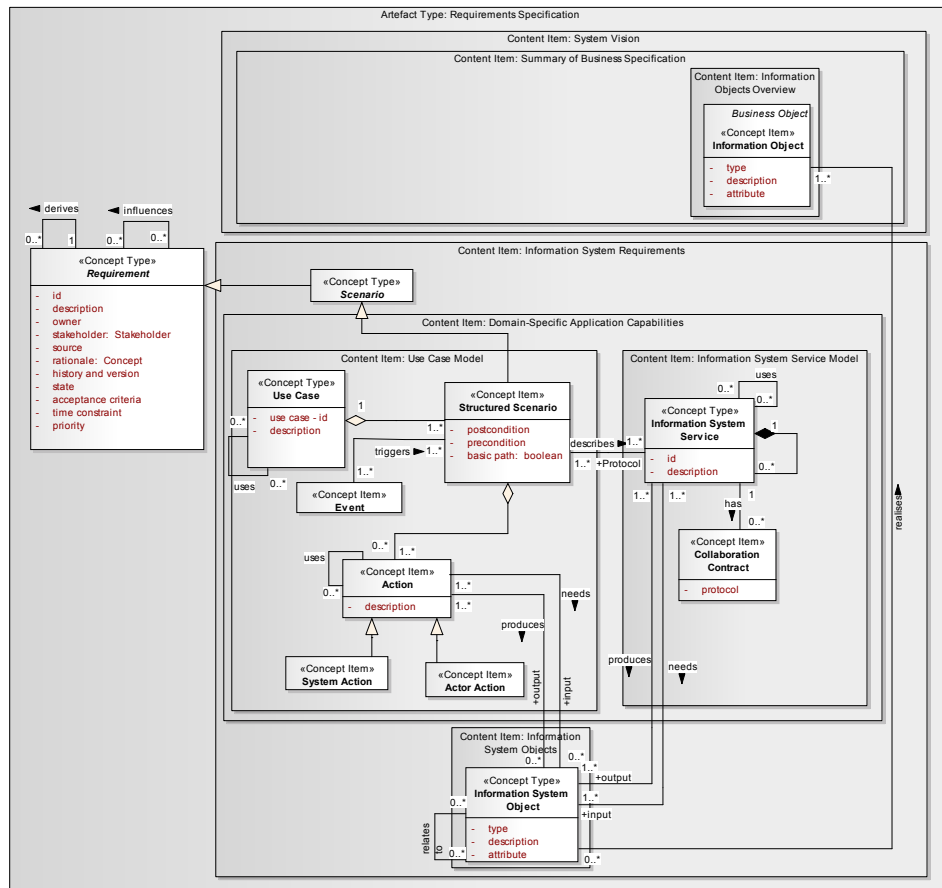


Figure 4.17: Excerpt of the concept model considering data modelling in the *Requirements Specification*. The upper part illustrates the dependencies to the corresponding concepts of the business specification, captured in the *System Vision* (see also Fig. 4.12).

As depicted with the concepts on the upper part of the figure, an actor realises in the requirements specification either a user group or an external system, both defined in the system vision. We allow to realise several user groups or systems with one actor. The reason is that user groups are defined by their characteristics within an organisation (their abilities, their needs, and their relation to certain business domains), while actors are defined and structured by the functions of a system that they can execute.

4.5.2.5 Concepts for Modelling Quality

Figure 4.19 illustrates a set of self-contained concepts for modelling quality. We refer in particular to *Generic Scenarios*, to *System Quality Requirements*, and to *Quality of Service*. We put emphasis on quality aspects that relate to demanded system properties and omit for reasons of complexity concepts to specify architectural constraints.

Generic Scenarios. With scenarios, we make no assertion on a particular requirements class, as they cover functional and non-functional aspects. For this dif-

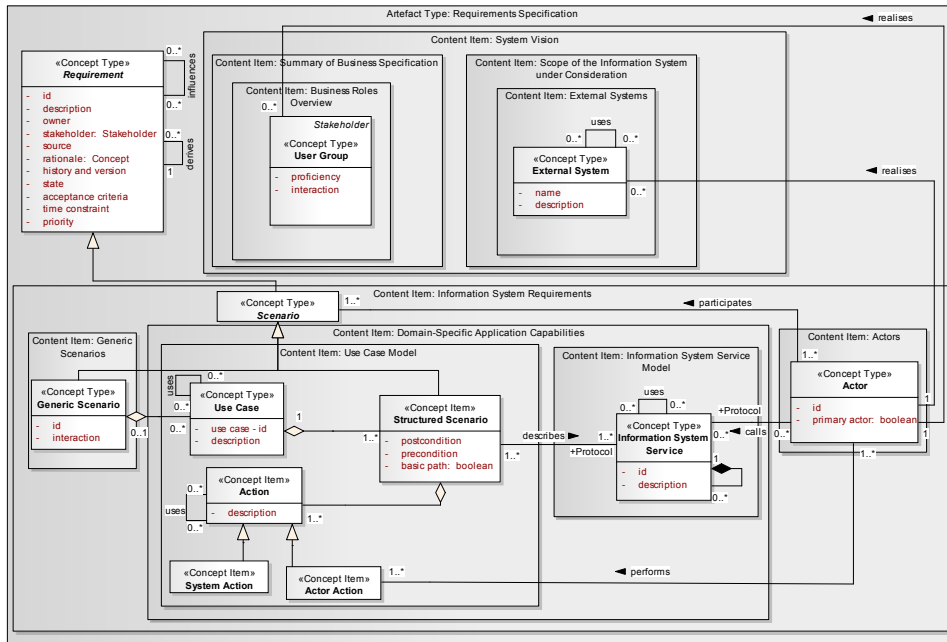


Figure 4.18: Excerpt of the concept model considering modelling of actors in the *Requirements Specification* (see right part of figure); The left part illustrates the dependencies to the concepts for modelling system behaviour (see also Fig. 4.16). The upper part illustrates the dependencies to the concepts for modelling business roles, captured in the *System Vision* (see also Fig. 4.13).

ferentiation, we specialise the concept to structured scenarios as part of functionally motivated use cases and to generic scenarios, used to express non-functional aspects. We use generic scenarios to informally describe sequences of interaction between actors and any facet of the system or its environment. These scenarios have a non-functional character, because they do not illustrate how the functionality of the system directly supports a user group in performing his business task. Hence, in contrast to the scenarios of a use case, generic scenarios do not include a set of (system) actions relating to information system objects.

The concept type is motivated by the need of

1. describing interactions / behaviour that has to be *prevented*, e.g., defined with attack scenarios via misuse cases [HP08, Ale03, WMFIL09]).
2. describing interactions / behaviour that directly has to be *supported* without any relation to business processes, e.g., by describing maintenance tasks w.r.t. modifications of the source code as proposed in ATAM [BCK03] (Architecture Tradeoff Analysis Method).
3. structuring, ordering and grouping use cases emphasising what users have to perform *before executing* a use case or *between executing* several use cases.

The activities described with generic scenarios cover an assessable view onto system quality requirements and architectural constraints, whereby both concepts have a satisfy-association to the generic scenarios.

System Quality Requirements. As discussed in Sect. 4.2.2.2, we conceptually express quality over three levels of abstraction, covering IT-related goals that relate to quality attributes (introduced in Sect. 4.4.2.4), system-supported activities expressed via generic scenarios, and system quality requirements.

4.5 Artefact Type: Requirements Specification

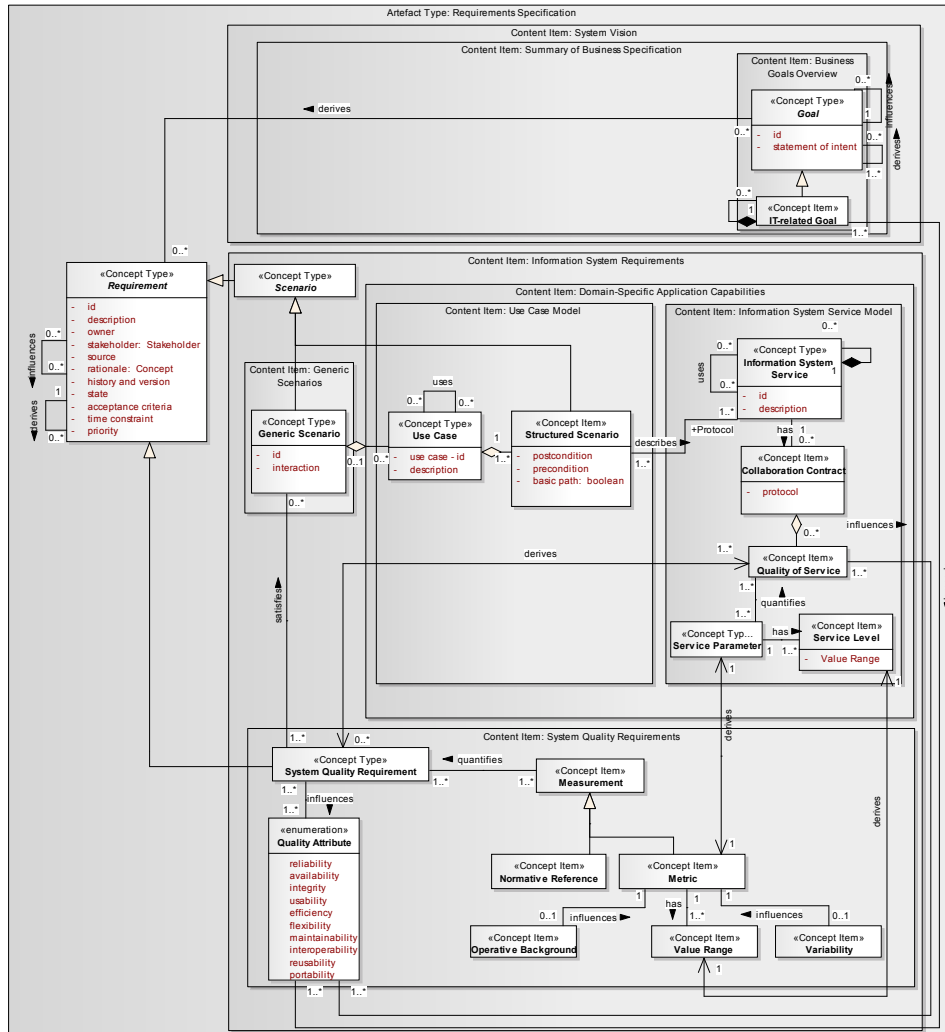


Figure 4.19: Excerpt of the concept model considering modelling of system quality requirements and quality of service in the *Requirements Specification*. The middle part illustrates the dependencies to the concepts for system behaviour modelling (see also Fig. 4.16). The upper part illustrates the dependencies to the concepts for modelling business goals, captured in the *System Vision* (see also Fig. 4.13).

We use system quality requirements to constrain a system and its (development and usage) environment in order to support an activity and, thus, to satisfy a goal. System quality requirements constrain specific behavioural and structural elements⁵ of the system and its environment in their properties and conditions by the use of measurements. The achievement of those properties supports the activities that rationale the requirement. The concept relates to the one of quality attributes, because the requirements influence specific quality characteristics that the stakeholders perceive when using the system.

For specifying system quality requirements in a quantified or at least assessable

⁵ An example for constraining a behavioural element could be the demanded response-time of a search function. An example for a constrained structural element could be the demanded structure of the documentation.

manner, we use as a measurement either *Normative References* or a concrete *Metric*. A normative reference is a referenced guideline describing properties of system elements, such a norm for web accessibility or a user interface design guideline. A metric is a non-ambiguous measurement for a test, such as “response-time in seconds”, “mean-time to failure (MTTF)”, “number of jobs per minutes”, or “existence”. For each metric, we define at least one *value range* and a *variability*. The variability represents circumstances and a range in which actual measurements may vary from an expected value or a probability in which the measurement may fail (useful if referring to unknown technologies or if depending on external suppliers). Finally, the *operative background* gives additional context information, such as expected transaction loads, storage loads, or the number of expected parallel executions of certain functions

Quality of Service. So far, we described the concepts for modelling quality by referring to scenarios. An additional means to define quality requirements is given in the context of information system services. As already mentioned, there exists the possibility of defining for each service a collaboration contract. A collaboration contract describes expected characteristics of an information system service including also expected quality of service.

The quality of service (or “service level agreement”) defines according to the *IT Infrastructure Library* (ITIL) [oGCO07] a contractual agreement on a set of service parameters including a set of service levels. Both are means to demand for a quality attribute for the service. A service parameter is a concrete measurement over the service life cycle, and a service level includes the expected value range for the chosen measurement, (like 96% availability). Both the service parameter and the service level quantify the quality of service to a measurable level and relate to the measurements given in the context of system quality requirements.

From a methodological point of view, system quality requirements can be inferred from the quality of service, the metric from a service parameter, and the value range from a service level (and vice-versa). The quality of service, however, covers only a subset of the possibilities given by system quality requirements, because the quality of service is restricted to user-visible characteristics that are observable when calling a service. Internal characteristics, e.g., maintainability-related ones, are not in scope. Further information on the conceptual differences between system quality requirements and the quality of service can be taken from the contribution from Brenner et al. [BGN06].

4.5.3 Syntax

In Tab. 4.4, we summarise those description techniques to which we potentially refer for representing the content of the requirements specification. Where reasonable, we give criteria for variations, similarly as done for the syntax of the business specification.

Regarding the content items in the system vision that summarise the content of the business specification, we distinguish, for example, between the use of tools relying on a unified model and the creation of the artefacts without tool support. If referring to tool support, the same content created for the business specification can be recorded within the system vision while supporting, due to the use of the same model basis, consistency between both (e.g., during modifications).

4.5 Artefact Type: Requirements Specification

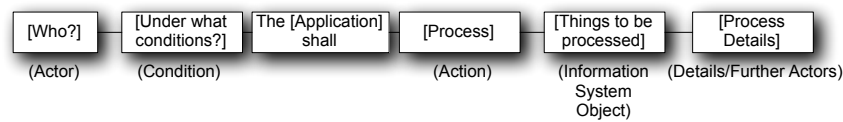
Table 4.4: Representation of content items of the requirements specification.

Content Item and Concepts	Variation Criteria	Description Techniques
Summary of Business Specification	No tool support	Informally in natural language via table
	Tool support	Reproduction of envisioned model excerpts
	None	Context diagram, optionally supported by rich pictures
	None	Informally in natural language via table
External Systems	None	Use case overview diagram as proposed by the UML [OMG10a, OMG10b].
Information Sys. Service Overview	None	Graph or in natural language, structured, e.g., by a table
Features	None	Informally in natural language via table
Use Case Model	Low complexity	Informally in natural language via templates as given by Cockburn [Coc00]
	High complexity	Diagrammatic representation with data flow-based techniques (like Message Sequence Chart (MSC) [IT99]) or with control flow-based ones (like UML Activity Diagrams [OMG10a, OMG10b]); See also the techniques for business processes in Sect. 4.4.3
Information System Service Model	None	Event/Response Tables [Wie03], Input/Output Tables [HT09]
Functional Requirements	None	Temporal logic, as given by goal-based approaches [vL03]; In case of natural language, via structured pattern-based text construction as shown by Fleischmann [Fle08] and Rupp [Rup07].
Actors	None	Informally in natural language via (actors) list
Generic Scenarios	None	Informally in natural language structured via templates (like story cards known from agile development process models)
Information System Objects	Low complexity	Informally in natural language (via tables)
	High complexity	UML class diagram [OMG10a, OMG10b]
System Quality Requirements	None	Informally in natural language; When describing the component architecture via rich components [Dam05, DVM ⁺ 05]
Architectural Constraints	None	Informally in natural language
Deployment Requirements	None	Informally in natural language
Migration Requirements	None	Informally in natural language additionally supported with tables for mapping of information system objects and pseudo code for data conversion algorithms
Project Requirements	None	Informally in natural language additionally supported by diagrammatic representations for the description of time schedules, e.g., by gantt charts
Obligations	None	Informally in natural language
Requirements Risk Status Report	None	Informally in natural language via table
Glossary	None	Informally in natural language via table

Similar as it is the case for the representation of the business specification, however, many concept types remain to an informal representation in natural language. Nevertheless, there are concepts that allow their representation with structured pattern-based text reducing the risks of linguistic defects (see also Kof [Kof05]).

An approach that proposes such a pattern-based specification of requirements is given by Fleischmann [Fle08]. Such approaches refer, in particular, to the concepts used to model behaviour, as they rely on well-understood key concepts, like conditions, stimuli and responses (see Sect. 4.5.2.2). Example 4.8 illustrates a possible pattern to structure natural language according to the single concepts given for modelling behaviour.

Example 4.8. *Template for Structured Text at the Example of Functional Requirements*



4.6 Artefact Type: Traceability Matrix

The traceability matrix builds an own artefact type that captures the project-specific content dependencies between the content items in the business specification, in the requirements specification, and in further artefact types of subsequent development activities. Hence, the traceability matrix has no own concept model, but reproduces selected associations between the concept types defined in foregoing sections and ones to be provided by the artefact model into which the concept model is (process-)integrated.

We already introduced in Sect. 4.3.3 an exemplary set of content dependencies that also are of interest when defining the traceability matrix; for instance

- the *satisfies*-dependency between a business process and a process-related goal enabling the determination of the rationale for a process model, or
- the *realises*-dependency between the actions of a use case scenario and the process steps to be executed in a system-supported manner enabling backward tracing from the concepts in the requirements specification to ones in the business specification.

However, since the dependencies to be reproduced as part of the traceability matrix arise from the project-specific exemplars of the artefact model at hand⁶ and depend on the artefact model provided by the overall development process model, we do not introduce a structure model, neither do we introduce a pre-defined set of associations for a content model.

Furthermore, note that the definition and the management of a (project-specific) traceability matrix depends not only on the project-specific selection of the dependencies to be managed in the matrix, but also on the used tool infrastructure. The traceability matrix can, for example, be manually generated and managed in a spreadsheet or it can be generated and managed by reproducing selected associations in a tool-supported manner. Which of the variants to choose in a project

⁶ More precisely, the possibilities of reproducing the associations arise from the possible and necessary degree of completeness in the artefacts, see also Sect. 4.2.3.2.

4.7 Artefacts as Interfaces for a Process Integration

depends also on the necessity to keep the dependencies in the matrix consistent and the effort needed to manage the matrix.

A discussion on the different options for representing and managing the traceability matrix can be taken from our previous contribution [MFK09].

4.7 Artefacts as Interfaces for a Process Integration

In Sect. 2.2.4, we gave an introduction into the boundaries of RE considering the relation to chosen activities of the development life cycle. For this, we took into account the contributions of Damian et al. [DC06] and of Hood et al. [HWFP07]. Both give a generic overview of basic activities that potentially make use of domain-independent requirements artefacts.

In this section, we now can relate the activities of the development life cycle to specific (domain-specific) content items and, thus, show which of the content items have to be considered

- for establishing, during process integration, the associations between the BISA-relevant content items and further ones given by the development process model into which we integrate the BISA reference model.
- during customisation of the reference model at project level when reflecting on potentially underspecified artefacts and their effects on development activities that rely on those artefacts.

Please note that the activities, that relate to the content items, have to be provided by the development process model into which the artefact model is process-integrated⁷, whereby we stick in the following to an abstract description of the activities as done in Sect. 2.2.4.

Design & Implementation. *Information System Requirements* define the demanded systems functionality and quality whereby the content item serves as an input for design activities. We refer with *design* to the design of the logical architecture and to the one of the technical architecture. The latter is in scope of *System Quality Requirements* and of *Technical Restrictions* (found in the content item *Architectural Constraints*). Regarding the relation to implementation, *Business Restrictions* have to be taken into account, as they potentially define implementation guidelines and coding standards. Finally, the content item *Business Domains* relates also to design activities, because business domains can be taken as orientation to structure the logical component architecture in a business process-oriented manner.

Quality Assurance. With quality assurance, we refer to analytical quality assurance that aims at discovering defects in the product or the process (including produced specification documents). This defect detection is typically done by means of tests, inspections, or reviews as part of, e.g., quality gates [Wag07]. Regarding activities performed during tests, the performance of acceptance tests is of interest, since these activities directly refer to the requirements specification checking whether the system under consideration meets the requirements. Hence, the creation of acceptance tests depends on *Information System Requirements* with a particular focus on the *Use Case Model* and the *Information System Service Model*. In addition, the artefact type *Traceability Matrix* supports testing activities, because it relates the requirements to the (business) goals and to the content of the system specification.

⁷ In Sect. 6.2, we perform a concrete process integration into the V-Modell XT and refer to the artefacts and activities provided by the V-Modell XT.

Risk Management. While risk management aims, in general, at the detection, treatment, and prevention of risk factors of a project, we distinguish between risks that affect a customer (like a market rejection of a business service) and ones that affect the development (like unfeasible requirements). The first is in scope of the *Business Demands Analysis*, the latter is in scope of the *Requirements Risk Status Report* that summarises defected requirements. We additionally take into account the identification of moving targets as a risk factor arising from volatile requirements. The detection of moving targets and their prevention is supported by the *System Vision* and by the *Traceability Matrix*.

Project Management. We consider, in context of project management, general activities performed during acquisition and bidding procedures, the definition of contracts, the performance of complexity calculations and cost estimations, and the general project planning.

Regarding acquisition and bidding procedures, the *Business Vision* and the *System Vision* are of interest, since these items specify the main objectives of the project and the major features of the system. A similar support is given for contract management. In particular, the *Features* in the system vision support the creation of (frame) contracts, because features give an abstract view onto an amount of requirements while also giving the possibility of coupling penalty agreements to those requirements. Regarding the performance of complexity calculations and the cost estimations, we refer, if performed via function points, to the *Use Case Model (and the Overview)* and to *Generic Scenarios*. Finally, project planning, including the definition of team structures, work packages, planning of milestones, and time schedules, are directly supported by the *Project Scope* in the business vision and by the *Organisational Requirements*. In addition, *Business Domains* support the approval activities, e.g., by structuring workshops according to single business domains and related stakeholders (see also the process model in Sect. 4.8).

Configuration & Change Management. The identification of defects and potentially resulting change requests is supported by the *Traceability Matrix*, as it supports impact analyses. We additionally take into account the dependency to the *System Vision*, since it serves the identification of moving targets and, thus, the evaluation of potential changes (and identification of potential problems).

Release Management and Deployment. The release and deployment strategy of a system under consideration has to be performed in compliance with *Integrational Requirements* whereby this content item serves as an input for corresponding activities.

4.8 Process Model

In the following, we introduce the process model of BISA, i.e., the entities necessary to define an artefact-based process. We rely on the concepts defined by the meta model in Sect. 3.3 and define the milestones, the activities and tasks, and the roles being coupled to the artefact model. As a first step, however, we give an overview of the ingredients of the process model and the necessary principles to enable the flexible customisation.

4.8.1 Overview of Process Model

According to the meta model, the process model consists of phases, activities (including tasks), and milestones. The phase *Business Information Systems Analysis* has already been introduced in Sect. 4.1 as a domain-specific interpretation of RE. We sketch with Fig. 4.20 an overview of the process elements of this phase.

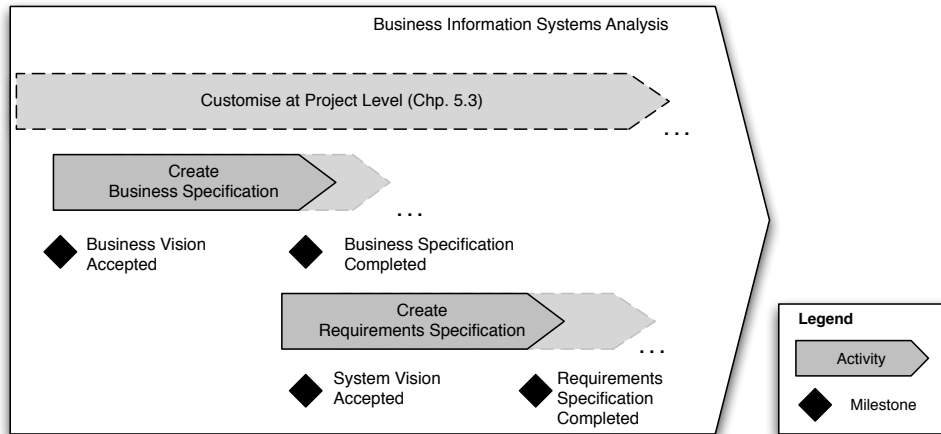


Figure 4.20: Overview of process elements for Business Information Systems Analysis.

We distinguish two activities for creating the introduced artefact types *Business Specification* and *Requirements Specification*, while each of the activities include a set of tasks to create the single content items. These tasks are defined in Sect. 4.8.4. Since the traceability matrix is an artefact type whose content creation accompanies the overall development life cycle, corresponding process elements are not in scope of the phase. For the same reason, we exclude (requirements) change management tasks from the process model. In addition, we depict another activity considering the customisation during the content creation. The customisation accompanies the creation of the artefact types in dependency to individual project characteristics and is defined in Chp. 5.3. In order to enable the customisation following the artefact-based philosophy, we discuss in subsequent Sect. 4.8.2 the necessary properties of the process model.

In Fig. 4.20, we illustrate furthermore four milestones, which we define in Sect. 4.8.3. Each of the activities indirectly relates (over the artefacts to which they are coupled) to two milestones. One milestone defines, when instantiated, when to accept the corresponding visions, another milestone defines when to complete the whole artefact type.

4.8.2 Properties supporting Customisation at Project Level

We now introduce two properties, which support a flexible customisation: an iterative content creation and the definition of the process model in a generic way. Both are explained in the following.

Iterative Content Creation. As already described in Sect. 2.2.1.1 in the fundamentals and reflected in the definition of the phase in Sect. 4.1, the content creation is cyclic. Figure 4.21 illustrates in its bottom part, according to Conklin et al. [CW97], the principles of an iterative problem statement and problem solving (with the dotted lines) and on its upper part the interpretation for BISA.

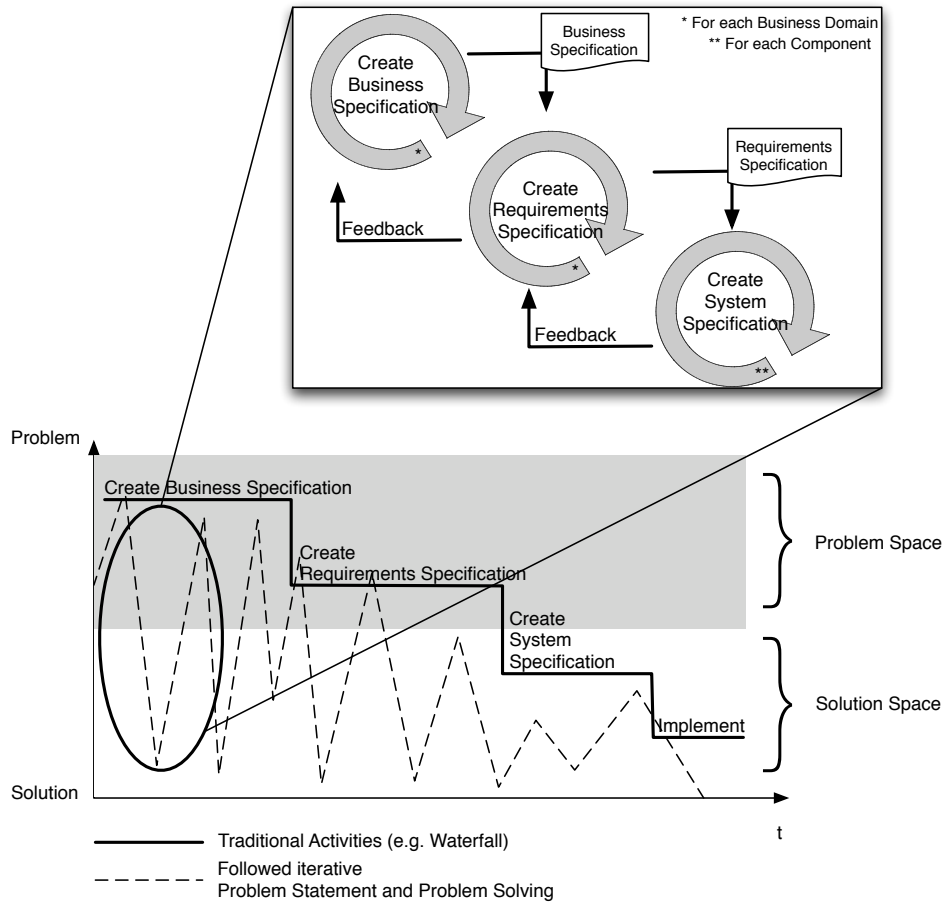


Figure 4.21: Iterative content creation of the artefacts structured via business domains.

As a matter of fact, an efficient process relies on the need of continuously narrowing down the stated problems in parallel to a growing knowledge on the ideal solution. This means that we need the possibility to step-wise reason on acquired requirements, modify existing ones, and discover new (hidden) ones based on the feedback arising from each of the made design decisions (see also [vL08]). Thus, the approach is neither purely top-down nor is it bottom-up. To enable this iterative approach, we make use of business domains to structure during customisation decision taking. We make use of business domains to structure the approval process, e.g., by organising workshops with the stakeholders according to the business domains to which the content to be negotiated belongs⁸ and by deciding the necessary and possible degree of completeness for the content of each of the business domains (see Sect. 5.3 for further information). However, how exactly to execute the process depends on the provided development process model (see Sect. 5.3.2.2).

Genericness of Process Model. Although aiming at an iterative creation of the artefacts, the provided process model itself is generic in two ways. First, we do not define any relations between the activities or between the tasks. The order in

⁸ See also appendix B.1 that illustrates different possibilities of structuring the (contents of the) artefacts according to business domains.

4.8 Process Model

which to create the artefacts exclusively arises from the agreement on the artefact structure to be delivered and the chosen milestones (coupled to the artefacts). Both the artefact structure and the milestones are customised when initially setting up the project (see Sect. 5.3.3.1). Second, the provided method structure is kept flat and does not take into account a particular syntax and, thus, it supports creativity for producing the artefacts. See also the meta model in Sect. 3.3.2.4 for further discussions.

4.8.3 Milestones

Figure 4.22 illustrates the realisation of the milestones concept and shows to which parts of the artefact model the milestones are coupled via directed associations.

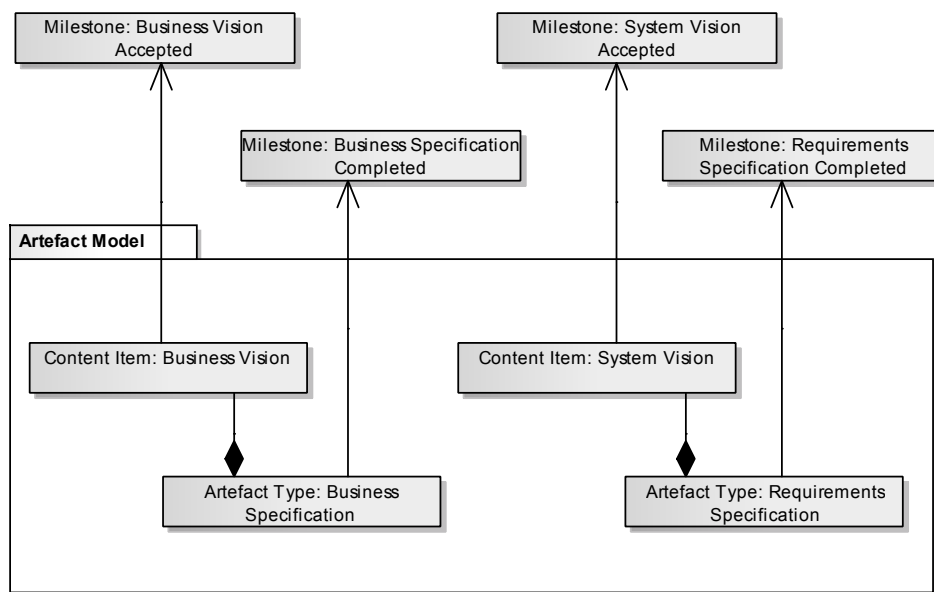


Figure 4.22: Milestones in relation to artefact model.

The milestone *Business Vision Accepted* relates to the business vision and defines the point in time when the business vision shall be accepted. The milestone *Business Specification Completed* describes the point in time when the business specification shall be completed, i.e., when the final state (*Finished*) shall be reached for the overall artefact type (see the state model in Sect. 4.3.4). Same is done for the requirements specification.

The definition of two milestones for each of the artefact types enables the definition of a point in time when to initiate the elaboration of the artefact in full (when the steering objectives captured within the visions are clear) and when to finish the artefact. Thus, we are able to execute the process within a structured planable process frame.

4.8.4 Activities for Content Creation

We now summarise the tasks of the two activities considered by BISA. These tasks comprehend several content items and result from the feedback collected during the elaboration of the process model in an industrial environment (see Sect. 3.3.3).

4.8.4.1 Activity: Create Business Specification.

Table 4.5 on page 148 summarises the tasks for creating the business specification. For each of the tasks, we describe the content items that serve as an input (or further informal sources) and the items that are produced or modified, while the input/output-relations result from the dependencies in the concept model. After defining the current state of an organisation (current business processes and services), we perform a gap analysis to sketch a business vision. The goals are decomposed in order to steer the decomposition of the business processes into a desired direction. In particular, the decomposition is performed for business processes (and services) that directly contribute to the satisfaction of the goals (the core processes). Based on the idealised business processes, the structure (business process hierarchy) is re-designed.

4.8.4.2 Activity: Create Requirements Specification.

Table 4.6 on page 149 summarises the tasks for creating the requirements specification following the same structure as done in the foregoing activity. The first task considers the definition of the system vision. The system vision can be defined when the business specification has a sufficient degree of completeness, i.e., when the business task descriptions are available. These business tasks serve as an input for defining the scope of the envisioned system. When the system vision is defined, the subsequent tasks consider the specification of the requirements, including

1. the elicitation of the requirements,
2. the validation of the requirements, and
3. the classification of the requirements into corresponding classes (according to the concept model) in dependency to their assertion.

4.9 Role Model

We distinguish two (primary) roles with corresponding abilities and responsibilities: the *Business Analyst* and the *Requirements Engineer*. Indirectly participating roles, such as a test manager, are omitted, because they have to be provided by the development process model into which the BISA reference model is integrated.

Business Analyst. The business analyst has the responsibility for the business specification and the activity considering its creation. He is the primary contact for the stakeholders w.r.t. the content of this specification. This role is characterised by domain knowledge on a certain industrial sectors, like the financial sector or insurance, including typical business processes in such industrial sectors.

Requirements Engineer. The requirements engineer is responsible for aligning the business needs with the needs towards corresponding information systems and, thus, the role has the responsibility for the requirements specification and the related activity. The requirements engineer needs a special consideration, because this role builds the bridge between business analysts, system architects, project managers, and any further roles that indirectly participate in the process. In contrast to the process model, we consider with this role also the responsibility for the traceability matrix, as we see the requirements engineer to be responsible to realise and control the specified requirements (although the creation of the matrix depends on the outcome of other activities).

Table 4.5: Overview of tasks of the activity *Create Business Specification*.

Task	Output Items	Input Items	Description
Define business context	Business context	Business cases, strategy papers, policies	Identify and describe objectives and principles
Derive business goals	Business goals	Business context	Identify business goals on the basis of the mission and value chain
Specify restrictions	Business restrictions	Policies	Identify business rules and constraints
Identify current business processes and business services	Business capabilities	Business cases	Identify as-is business processes (without process steps) and services
Perform gap analysis	Project scope	Business goals, business capabilities	Investigate if current state matches with defined principles and identified business goals
Complete business vision	Business vision	Elaborated content items	Complete, based on the defined project scope, the business vision and approve it.
Perform stakeholder analysis	Business roles	State charts, workshops	Identify stakeholder, process owner and user groups related to the project scope
Define process-related goals	Process-related goals	Business goals, principles	Define process-rel. goals that satisfy business goals
Categorise business processes	Business process model	Business services, business goals	Categorise processes based on how they contribute to achieve business goals and services
Decompose relevant business processes	Business processes and business tasks	Process-related goals, workshops	Define work flow within the business tasks for identified core processes
Define business information model	Business information model	B. processes and b. services	Infer business and information objects
Define business domains	B. domains	B. roles, b. capabilities, b. objects	(Re-) define business domains
(Re-) design business services and organisation structure	Organisation structure, business services	Business processes	Adjust structure according to ideal business processes
Define needs for automation	IT-related goals and costs for automation	Business processes and services, business cases	Identify what business services and processes need to be supported by information systems
Rationalise decisions	Business demands analysis	Identified risks, value and cost for each b. service	Complete business demands analysis
Specify glossary	Business glossary	Workshop protocols	Define domain-specific terms in glossary

Table 4.6: Overview of tasks of the activity *Create Requirements Specification*.

Task	Output Items	Input Items	Description
Summarise business needs	Summary of business specification	Business specification	Summarise content of business specification being in scope of the system
Scope system's capabilities	Use case overview, IS service overview	B. services and b. tasks	Identify candidates for use cases and services
Scope system's boundary	S. overview, ext. systems	—	Define interrelation to operative environment
Complete system vision	System vision	—	Complete vision (e.g. by additional features) and align with business specification
Elicit and prioritise requirements	abstract requirements	—	Elicit initial requirements and prioritise
Analyse user interaction	Use case model, actors, information system service model, Generic scenarios	Business capabilities, business rules, IT-related goals, workshop protocols	Define expected (functional and non-functional) user interaction and services
Derive functional requirements	Functional requirements	Use case model, workshop protocols	Define functional requirements not covered by use cases
Define information system objects	Information system objects	IS services, use case model	Define processed IS objects in relation to information objects
Specify system quality requirements	System quality requirements	Gen. scenarios, IT-related goals, QoS	Define system quality with measurements
Define architectural constraints	Architectural constraints	Generic scenarios, business constraints	Define logical and technical architectural constraints
Define organisational requirements	Organisational requirements	Workshop protocols, bidding documents	Define project requirements and obligations
Define integrational requirements	Integrational requirements	Project plan, information system objects	Define deployment and migration requirements
Validate requirements and assess risks	Requirements risk status report	Any requirement	Check requirements for defects and assess risks
Adjust priority	Any requirement	Any requirement	Adjust priority based on risks and costs
Specify glossary	Requirements glossary	Workshop protocols	Define technical requirements-related terms

4.9 Role Model

Artefact-Based Customisation

In subsequent sections, we construct the artefact-based customisation approach and transfer customisation principles, e.g., of situational method engineering, to our artefact-based philosophy.

After giving, in Sect. 5.1, an overview of the considered customisation stages, we introduce in Sect. 5.2 the approach for a process integration of the BISA reference model w.r.t. its structural properties defined with the meta model in Chp. 3. Having defined the approach for integrating the BISA reference model into a development process model, we construct in Sect. 5.3 the approach for a customisation at project level. For this, we describe how to (statically) generate exemplars of each sub-model of the reference model (artefacts, milestones, etc.) and how to (dynamically) create the content of the artefacts during project execution, both in dependency to project characteristics. Finally, having taken the view on the project-specific creation of the BISA artefacts (contents), we give guidance to analyse the effects in context of the overall project execution. We show the possibilities of reflecting on made decisions and the overall situation w.r.t. potentially under-specified artefacts and their effect on the development life cycle. On this basis, we support an overall evaluation for further going development steps. The content of this chapter is based, in part, on our contribution in [MFK09, MFLPW11].

At the end of this chapter, the reader will know the principles of artefact-based customisation and the shortcomings in activity-based approaches we can tackle. He will understand how to perform a process integration and how to systematically create the artefacts at project level whether in a solution-oriented or in a problem-oriented way, both in direct response to individual project influences. Chapter 6 then gives with two case studies evidence on applicability, advantages, and limitations of the introduced approach.

Contents

5.1	Customisation Overview	152
5.2	Process Integration	153
5.3	Customisation at Project Level	158

5.1 Customisation Overview

We introduced in Sect. 1.2 the objectives of the thesis consisting of the development of an integrated approach

1. that supports awareness of creating syntactically consistent and complete results for the domain of business information systems, and that
2. ensures flexibility during this creation.

In Sect. 2.6, we showed that there exist customisation approaches mostly incorporating the activity-based philosophy, e.g., in a development process model. Corresponding customisation approaches and the area of decision support systems propose means for the (static) selection and combination of methods according to project parameters when initially setting up a project. We also showed, however, that these approaches have shortcomings in supporting their process integration and in tackling the process variability in a project during set-up and during the dynamic execution.

In this chapter, we define the artefact-based customisation approach to perform a process integration of our BISA reference model and to customise the integrated model at project level by transferring given related work to the artefact-based philosophy. Figure 5.1 illustrates the customisation stages according to the scope defined in Sect. 2.6.1.

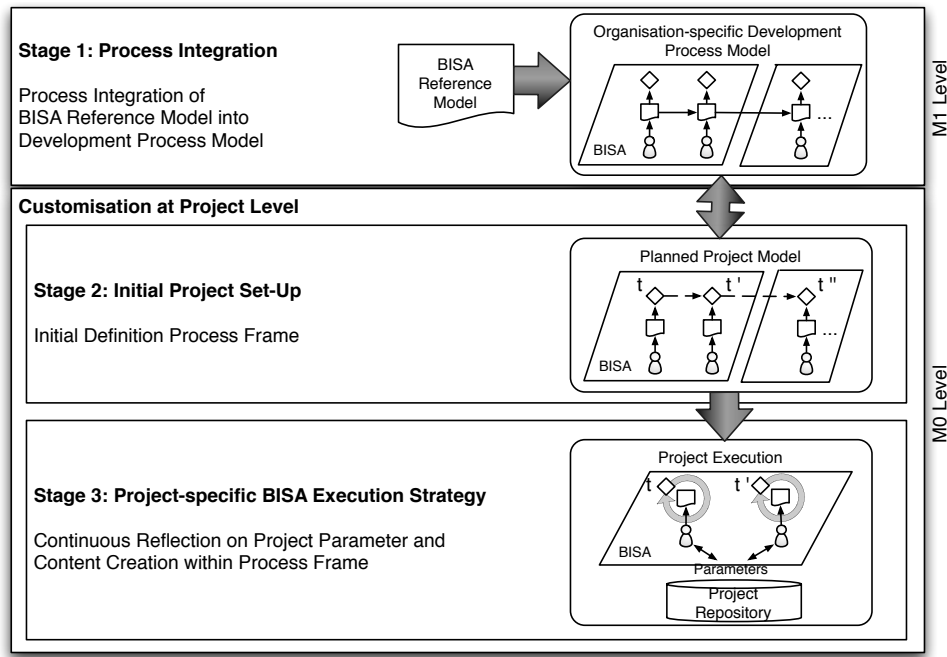


Figure 5.1: Overview of artefact-based customisation approach.

Process Integration of BISA Reference Model

The first stage represents the process integration of the artefact-based reference model of Chp. 4 into a development process model. To perform a process integration of the artefact-based reference model, we make, in particular, use of the structure model (in the artefact model) taking into account the interfaces for process integration defined in Sect. 4.7.

Customisation of BISA Reference Model at Project Level

Once we introduced the approach for integrating the BISA reference model into a development process model, we show in Sect. 5.3 the approach for its transparent and reproducible customisation at project level. For this, we transfer related work in the area of decision support systems complementing the area of situational method engineering to the artefact-based philosophy. We introduce a notion of project characterisation and project execution that relies on project parameters impacting the creation of the artefacts (instead of impacting the choice of methods). The actual customisation approach is then performed over two stages (see Fig. 5.1).

Initial Project Set-Up. The initial project set-up provides an approach for the creation of exemplars of the sub-models of the artefact-based reference model and thereby shows how to define a (planned) process frame.

Project-specific BISA Execution Strategy. The project-specific BISA execution strategy then results from the guidance of the content creation within the defined process frame in direct response to project parameters. This content creation considers the creation of the artefacts either in a problem-oriented or in a solution-oriented way. When reaching defined decision gates (milestones), we show how to reflect on the overall project characteristics, given by possibly underspecified artefacts and their impacts on further activities of the development life cycle that rely on those artefacts.

Setting in (Plan-based) Enactment. In Sect. 2.6.1.2, we distinguished between a plan-based enactment during set-up, i.e., the definition of a project plan that abstracts from the idealised project execution, and the actual project execution itself. In case of deviations in the real project execution to the planned one, feedback loops can be used to adjust the project plan. As also stated, however, the focus of the thesis lies on conceptualisation of decision support for a particular limited phase (BISA), and on the guidance of the content creation within that phase. Hence, we intentionally do not take into account deviations of the project execution to a planned one, and, thus

1. do not conceptualise the actual definition of a project plan, and
2. do not introduce mechanisms to distinguish a planned project and an actually executed one, nor do we introduce mechanisms to manage possible inconsistencies between both (see also the discussion of future work in Sect. 7.2).

5.2 Process Integration

In the following, we describe the approach for integrating the BISA reference model (defined in Chp. 4) into a development process model. For this, we recapitulate as a first step related work.

5.2.1 Recapitulation of Related Work

We showed in Sect. 2.6.2.1 that there exists no directly related work for a process integration, because such mechanisms are to be provided by those approaches that

5.2 Process Integration

shall be integrated. Kuhrmann et al. [KTF10], however, contribute an approach for adapting and integrating the V-Modell XT into an organisation¹ via four steps, each incorporating detailed instructions and experience-based practices.

Since the V-Modell XT and the artefact-based reference model have a similar artefact-based philosophy (see the construction of the meta model in Chp. 3), we can structure the approach for the process integration according to those four steps. In addition, we take into account our own development of the initial artefact-based approach and its process integration into the processes of the company Capgemini TS for the purpose of evaluation (illustrated in Sect. 3.3.3).

5.2.2 Approach for a Process Integration

For a process integration of the BISA reference model into a development process model, we analyse as a first step the situation and the resulting needs in the currently used development process model in a particular organisation. After gaining the necessary understanding of the content, the structure, and the application of the development process model in projects, the process integration is planned in a draft and, if necessary, approved with the corresponding process engineer. After completing the conceptualisation, the actual process integration is realised. Finally, the integrated model, respectively the modified development process model, is integrated again into the organisation.

Figure 5.2 illustrates these four steps for performing the process integration of the BISA reference model.

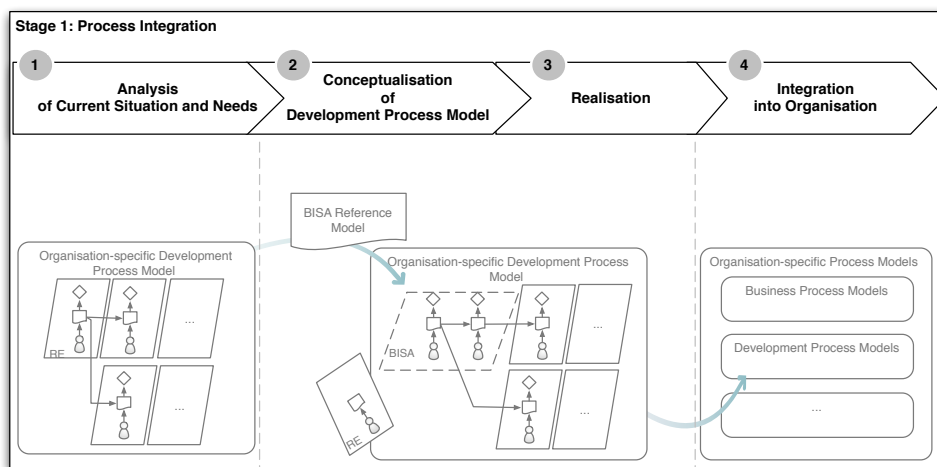


Figure 5.2: Overview of approach for process integration of the BISA reference model into a development process model.

5.2.2.1 Analysis of Current Situation and Needs

The analysis of the current situation and the needs serves to understand the possibilities and the necessities of process-integrating the BISA approach into a development process model of a particular organisation. This step includes

1. the analysis of the company-wide (reference) definition of the development process model (at M1),

¹ See also Sect. 2.5.3.2 and Sect. 2.6.4.2 describing corresponding related work.

2. the analysis of the project-specific application of the development process model and the inference of contemporary needs (at M0), and
3. a proposal of an integration strategy.

The field study presented in Sect. 1.1 represents such an analysis of current problems considering both the organisational and the project level. In the following, we describe the single steps of the analysis.

Analysis of Process Definition. The analysis of the process definition investigates the development process model itself and the meta model on which the development process model relies.

The investigation of the meta model serves to establish a terminological and structural mapping between the concepts used in the development process model and the ones used in our artefact-based philosophy (see our meta model in Sect. 3.3); for instance, the structure of artefacts with their relations to methods and how these concepts can be mapped. As already described in Sect. 2.6.1.1, a prerequisite for a process integration is that both meta models need the same or a similar philosophy, i.e., that the elements and relations defined in the meta model can be mapped on elements and relations given by the meta model of the envisioned development process model.

The development process model itself, serving in the organisation as a reference for development projects, then has to be investigated in order to infer an understanding of the relevant artefacts, milestones, roles, and activities (with tasks). Relevant study objects are general process descriptions, guidelines, or training material. A particular focus lies on descriptions of requirements engineering processes including business analysis processes.

Analysis of Project-specific Application and Inference of Needs. Once the development process model is understood, its application at project level has to be investigated considering initially produced requirements engineering artefacts. The purpose consists of

1. discovering mismatches between the recommended artefacts and the actually produced ones,
2. understanding the project parameters that cause those mismatches, i.e., that hamper the creation of the recommended artefacts, and
3. understanding the organisational culture respecting used methods and corresponding description techniques for producing the artefacts.

This analysis thus serves to understand the problems in the actual development process model and to collect an initial set of organisation-specific project parameters. A detailed description on possible research methods for conducting the analysis can be taken from the study design of our field study [MFWL⁺12]. These methods also include an economical investigation, which can be taken to analyse costs and benefits of a process integration.

Proposal of Integration Strategy. The last step in the analysis consists in the presentation and the discussion of the results with the project participants involved in the analysed projects, as well as with the process engineer being responsible for the development process model. This discussion clarifies the necessity of integrating the BISA reference model into the development process model, its approval, and how the integration shall be performed.

5.2 Process Integration

5.2.2.2 Conceptualisation of Development Process Model

Once the targeted development process model is understood and the process integration has been approved, it has to be conceptualised. With *conceptualisation*, we refer to a dry run and a continuous approval with the process engineer being responsible for the development process model.

The first step in the conceptualisation consists in the identification of the corresponding excerpt in the development process model.

Identification of BISA-relevant Excerpt. The results of the analysed projects, performed in the previous step, support the identification of typical scenarios in which requirements engineering is performed in projects. These scenarios provide an understanding of which artefacts are produced in which order (including corresponding milestones). The scenarios must include the creation of business process specifications and / or requirements specifications.

We do not, however, take into account comprehensive project execution strategies or project plan generation for the like (including strategies of arranging milestones for, e.g., several increments). Such comprehensive strategies must be provided by the development process model itself, respectively these are topics of particular management activities and not in scope of the model to be integrated.

Customisation of BISA-relevant Excerpt. Based on the identified excerpt of the development process model, it has to be customised according to the corresponding elements of the BISA reference model. For this, we consider a customisation of the sub-models introduced in Chp. 4 without taking into account project-specific specialities.

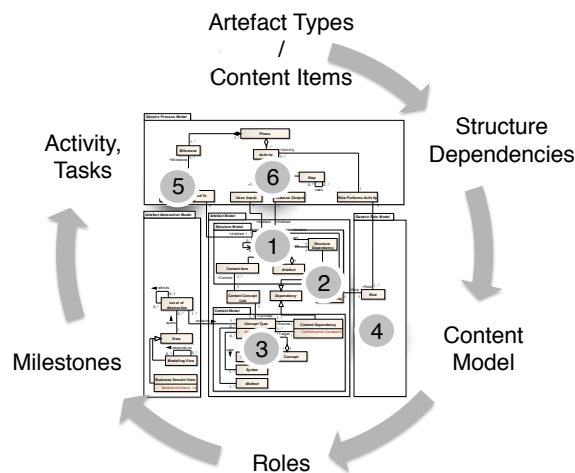


Figure 5.3: Order in which to integrate the elements and relations of the BISA reference model (view: meta model).

As shown in Fig. 5.3, we put during this integration the artefact model in the centre of our attention with a particular focus on the structure model and its dependencies. According to this view, we first integrate the artefact types with corresponding content items, establish the structural dependencies considering the interfaces defined in Sect. 4.7, and incorporate then the corresponding sub-models following the associations defined in the meta model.

We can define activities and tasks as a default recommendation that are preferred according to the organisational culture (being identified during the foregoing analysis) – if the elements and relations in the description techniques harmonise with the ones defined by the concept model.

During the customisation of the identified BISA-relevant excerpt of the development process model, however, we have to distinguish between

- the creation of new artefact types and associated elements,
- the replacement of existing artefact types and associated elements, and
- the modification of existing artefact types and associated elements.

Figure 5.4 depicts these three variations in a simplified manner.

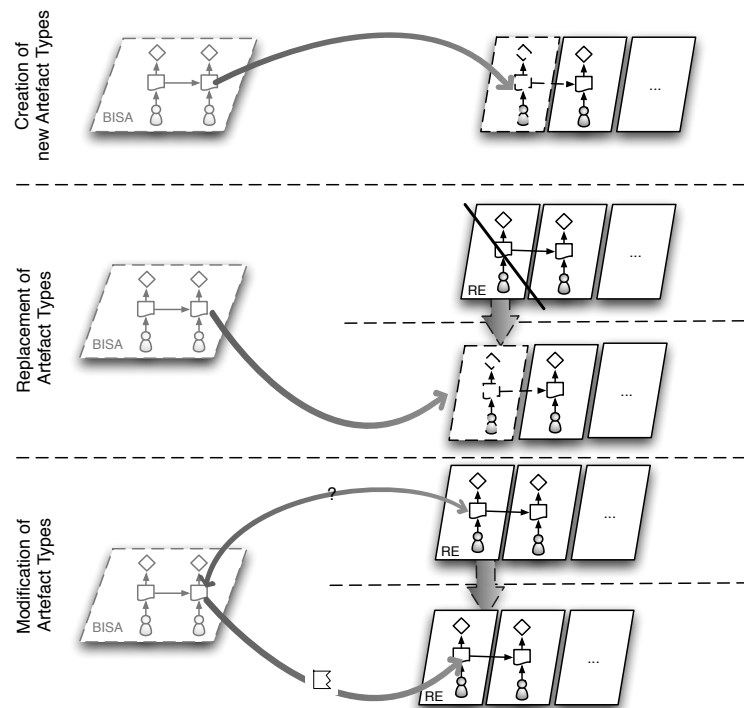


Figure 5.4: Variations for customising the development process model during the integration of the reference model.

In the first case of having to create new artefact types, e.g., if no business specification or equivalent artefact type is existent at all, we create this artefact type from scratch. In addition, we establish the associations between the new content items to further artefact types of the development process model whose creation rely on the new content items. To establish these associations, we refer to the interfaces for process integration defined in Sect. 4.7. The further sub-models, i.e., the roles, the milestones, and the activity with the tasks, are then coupled to the new artefact type as defined by the BISA reference model.

In the second case of replacing existing artefact types and relating sub-models, we remove the artefact type from the development process model, integrate the artefact type of the BISA reference model, and, where possible, take over the associations between the artefact type to be replaced and the further ones. The allocated sub-models (role model, ...) are then taken over the same way.

The third case considers the modification of existing elements in the development process model; for instance, by extending an existing artefact type “Requirements

5.3 Customisation at Project Level

Specification” with additional content items. Such an extension is only possible, if none of the content dependencies is violated. This means that, for each of the content items to be integrated (extended), it has to be possible to establish all the content dependencies of the included concept types to concept types included by related content items. The further sub-models are then similarly customised, e.g., by extending the abilities of existing roles.

Prerequisites for a Process Integration. Finally, the introduced steps assume that the development process model includes an artefact model and that this artefact model is defined via a structure model, as introduced in Sect. 3.1.3. Hence, both the existence of an artefact model and its definition via a structure model (potentially including a content model) are mandatory prerequisites for a process integration.

5.2.2.3 Realisation

Once the integration has been performed in a drafted, but comprehensive way, it is approved with the process engineer and realised. The realisation can be performed in several iterations, following the same procedure defined the section before (see Fig. 5.3 on page 156).

Because having defined in Chp. 4 two artefact types², we consider at most two iterations. Thus, the outcome of each of the iterations is a self-contained applicable set of artefact types with corresponding milestones, roles, and so on.

5.2.2.4 Integration into Organisation

There exists three possible integration strategies [KTF10]:

1. A “Big Bang” integration strategy, which announces the comprehensive new development processes model and declares it as a standard model to be used in those projects that begin after the announcement.
2. A step-wise integration strategy, which announces parts of the new development process model (e.g. the outcome of single realisation iterations).
3. An integration via pilot projects and training courses being given in those pilot projects.

The choice of the strategy depends on the organisational culture and the agreement during the last step of the initially performed analysis (see Sect. 5.2.2.1). At the example of the initially developed approach at Capgemini TS, we considered a step-wise integration and an integration via pilot projects and training courses. After working in the feedback from given projects, we declared the customised development process model as the standard reference model (see Sect. 3.3.3).

5.3 Customisation at Project Level

In this thesis, we aim with the customisation at project level at providing decision support on which artefacts (and contents) to produce. More precisely, the customisation approach shall

- support a *flexible* and reproducible creation of the artefacts in terms of guiding their creation in response to project parameters that affect the degree of completeness in the (contents of the) artefacts.

² The third artefact type *Traceability Matrix* is defined as an outcome of the content dependencies in the primary two artefact types and, thus, not directly to be integrated into a development process model.

- support *awareness* during the creation of the artefacts in terms of guiding the reflection on the impacts of potentially underspecified artefacts on further development activities that rely on those artefacts.

So far, we introduced the approach for the process integration of the BISA reference model of Chp. 4 into a development process model. In this section, we introduce the approach for customising the integrated model at project level in order to achieve the goals summarised above. For this, we briefly discuss as a first step the transfer of selected (activity-based) contributions and related principles to the artefact-based philosophy. Afterwards, we introduce the resulting principles of artefact-based customisation, before concluding with the description of the actual approach in Sect. 5.3.3.

5.3.1 Recapitulation of Related Work and Transfer

We already discussed related work in the area of customisation in Sect. 2.6. A particular focus lied on the approaches and principles given in following areas:

1. *Situational Method Engineering* [Bri96, Ode96, RDR03, BWHW05] contributing activity-based customisation mechanisms
 - a) for designing and persisting methods in a repository,
 - b) for characterising projects according to project parameters that argue for the choice of persisted methods, and
 - c) for assembling the retrieved methods in a project to enact a concrete project execution.
2. *(Content-centric) Decision Support Systems* [RPA⁺01, NTR05] complementing situational method engineering. Available approaches guide, by the use of project repositories, the classification, rating, and selection of a set of alternatives in the choice of methods (and description techniques) according to project parameters.

A discussion of corresponding approaches, their principles, and their limitations can be found in Sect. 2.6.5. In the following, we introduce how we transfer (activity-based) customisation principles, as incorporated by approaches related to the areas summarised above, to the artefact-based philosophy.

Artefact-based Decision Support. Based on the artefact-based reference model of Chp. 4 that serves as orientation to create domain-specific artefacts, we introduce a project repository as done in decision support systems. The necessity of introducing a method base is not given, since the artefact-based reference model itself abstracts from such a method base by making explicit implicitly necessary domain knowledge. In contrast to given decision support systems, however, we then characterise projects by the impact of project parameters on the artefacts, instead of indirectly characterising projects by parameters impacting the choice of methods for producing the artefacts. This characterisation is then accordingly done by means of the mentioned project repository complementing the artefact-based customisation approach. This supports the systematic creation of the artefacts in awareness of project parameters that directly relate to those artefacts.

Artefact-based Customisation Approach. As orientation for the actual customisation approach, we take the customisation approach defined by the V-Modell XT (see Sect. 2.6.4.2), because the V-Modell XT incorporates a similar artefact-based philosophy (see the meta model in Sect. 2.5.3.2). The V-Modell XT defines mechanisms for the initial project set-up by initially creating exemplars of the artefacts

5.3 Customisation at Project Level

and defining milestones, which are coupled to those exemplars. In contrast to the V-Modell XT, however, we also defined a domain-specific content model in addition to the (artefact) structure model. Hence, we can extend the customisation approach to decision taking and dynamic content creation during project execution considering a balanced problem orientation. This is enabled, in particular, because we introduced a notion of underspecified artefacts (def. in Sect. 4.2.3), which finally supports an appropriate decision taking w.r.t. the actual quality of the results.

5.3.2 Principles of Artefact-based Customisation

Figure 5.5 illustrates an overview of the basic principles of the artefact-based customisation approach. The left side of the figure illustrates the steps of the approach. The right side of the figure illustrates how the principles of decision support complement customisation following the artefact-based philosophy.

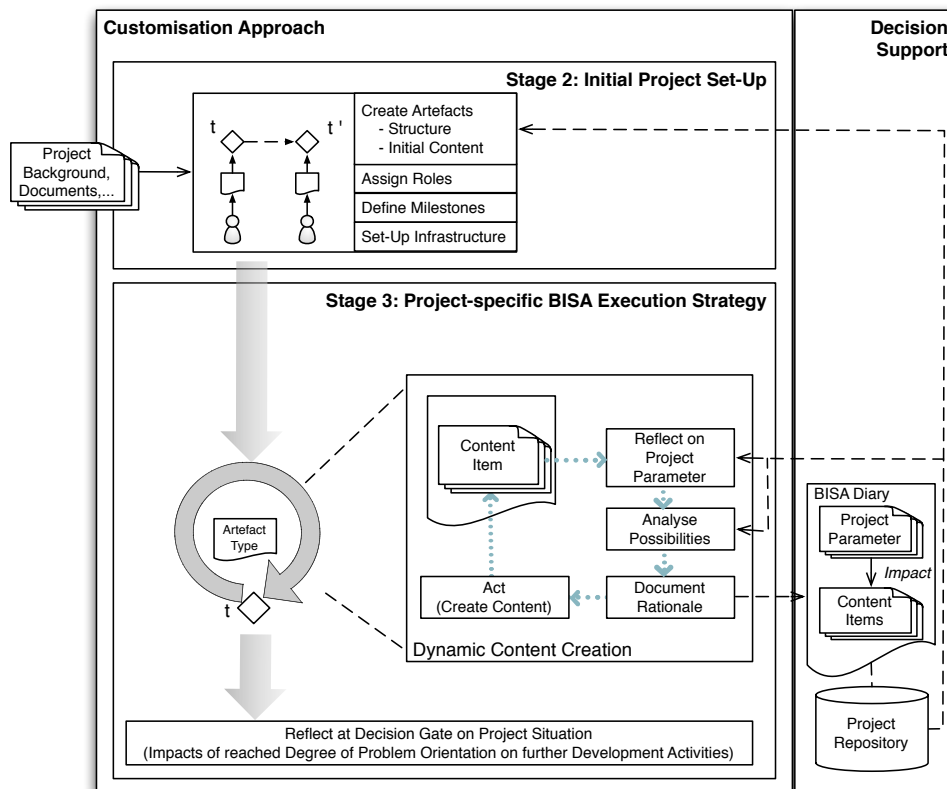


Figure 5.5: Overview of approach for customisation at project level (left side) in relation to project repository giving decision support (right side).

We introduce with a project repository a means to characterise projects according to project parameters impacting the creation of the artefacts' contents. The project characterisation is, in particular, given according to project parameters that argue for the creation of certain content items and ones that hamper their creation. Thus, we use the project repository and decision support, in general, as a means to achieve awareness of certain decisions to be made. To support the transfer of made decisions backwards from projects to the project repository, we introduce the additional artefact *BISA Diary*. This way, we can use a project repository in multi-

project environments to reuse organisation-specific experiences and decisions.

In a nutshell, we first set up a project, similarly as done in the V-Modell XT, by building exemplars of the defined artefact types, defining concrete milestones, assigning roles, and finally setting up the (tool) infrastructure. After creating the structure model, initial contents can be defined on the basis of given (project-specific) information. In a second step, we guide by use of the project repository the dynamic content creation towards a balanced problem orientation within the enacted process frame (created artefact structure and agreed milestones). When reaching the defined milestones during the content creation, we can show how to objectively reflect on the current project characteristics by taking into account the artefact abstraction model (Sect. 4.2) and our interpretation of underspecified artefacts leading to a solution-oriented process (Sect. 4.2.3). As the artefact-based reference model is integrated into a development process model, we can show how potentially underspecified artefacts impact further development activities that are associated with those artefacts.

In the following, we introduce the project characterisation and project execution in an artefact-based context, before introducing the actual customisation approach.

5.3.2.1 Project Characterisation in an Artefact-based Philosophy

In order to introduce the concepts of decision support following an artefact-based philosophy, we first need a notion of project characterisation. Based on this understanding, we can structure a project repository, the BISA diary, and finally build the customisation approach itself.

Figure 5.6 illustrates, on the right side, the necessary excerpt of our meta model in relation to the concepts used for project characterisation, situated on the left side.

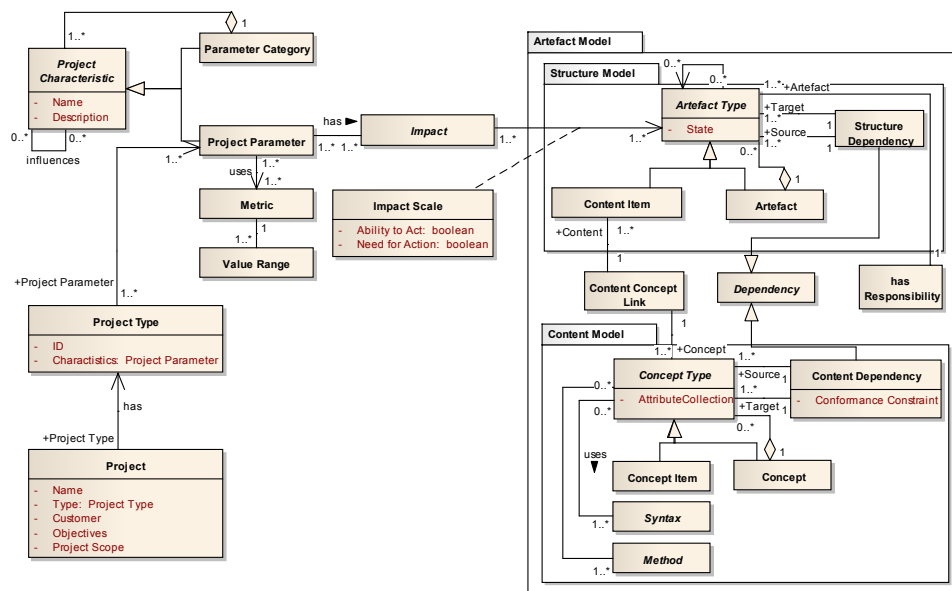


Figure 5.6: Project characterisation following the artefact-based philosophy reflected in project characteristics (on the left) and their impact on elements of the meta model for artefact orientation (on the right).

5.3 Customisation at Project Level

Project Characteristics. We characterise projects in general via *Project Parameters* that each has an impact on one or more artefact types or on selected content items with their underlying concept types. A project parameter is, in our context, an assessable condition from inside or outside a project influencing its execution; such as the availability of user groups made assessable via metrics and concrete values. The influence is expressed by an *Impact* (see the next paragraph). We see those project characteristics that are not assessable as *Parameter Categories*. Parameter categories can be decomposed to project parameters and, thus, serve as a means to group several related project parameters (like “Customer-specific parameters” grouping several assessable characteristics of the stakeholders).

Impacts and Impact Scales. Due to the artefact-based philosophy, we associate the impacts exclusively with the artefacts instead of associating them with methods. The formalisation of decision support is only suitable to a certain degree (see Sect. 2.6.3, [NTR05]). Many influences and related decisions cannot be formalised, such as budget that cannot be reasonably associated with particular elements of the artefact-based reference model. Therefore, we capture only those impacts in the project characterisation for which we can make unambiguous assertions and associate a project parameter by its *Impact* with (the creation of) an artefact. The definition of milestones and the assignment of roles are not in scope of a project characterisation and have to be performed by the project lead (or the corresponding role provided by the development process model).

Furthermore, we see the impacts to abstract in artefact-based decision support only from those aspects, which can be reasonably defined, and, thus refrain from abstracting from any possible human factors, expertness, experiences, political and strategic issues. Hence, we define two *Impact Scales* as a means to express how the project parameters take effect on the creation of the artefacts. They can either argue for the creation (“Need for Action”), or they can hamper the creation (“Ability to Act”).

Example 5.1. Impact Scales

An example for the impact scale “Need for Action” would be the existence of a governmental customer, which demands risk calculations thereby arguing for the creation of the content item “Business Demands Analysis”. An example for the impact scale “Ability to Act” could be that user groups are not available hampering the creation of the “Use Case Model” (and at the same time arguing for the creation of the “Information System Service Model”).

Projects and Project Types. So far, we have shown that projects are characterised by project parameters that impact via impact scales the creation of artefacts. We can now characterise a *Project Type*, being a generalisation of projects with same or similar characteristics, by a collection of chosen project parameters with same or similar values (see example 5.2).

A concrete *Project* is then consequently the instance of a project type. A project is a concrete set of artefacts being created by concrete roles over a specific time frame (in-between milestones), influenced by a collection of project parameters that characterise the project. As the artefacts are also created in response to particular objectives of a customer (and related stakeholders) and the project scope, we capture

the context information *Customer*, *Objectives*, and *Project Scope*³. For same reason, we also see the impact of project characteristics on the artefact types to be relevant for a project, rather than for a project type.

Example 5.2. Project Types

An exemplary project type “Small Scale Development Project” could be expressed by following project characteristics:

- Team size < 10 team members
- Degree of distribution: none, team members situated at customer’s company
- Existence of scheduled jour fixes with customer

5.3.2.2 BISA Reference Model in Context of an Artefact-based Project Execution

With *Project Execution*, we refer to the actual creation of chosen artefacts, i.e., the execution of the activities being coupled to the artefacts within a time frame. We consider in general three different variations of a project execution from the perspective taken by BISA. Figure 5.7 illustrates those variations showing both artefact types *Business Specification* and *Requirements Specification*, and a set of artefact types in the bottom representing, e.g., system design specifications.

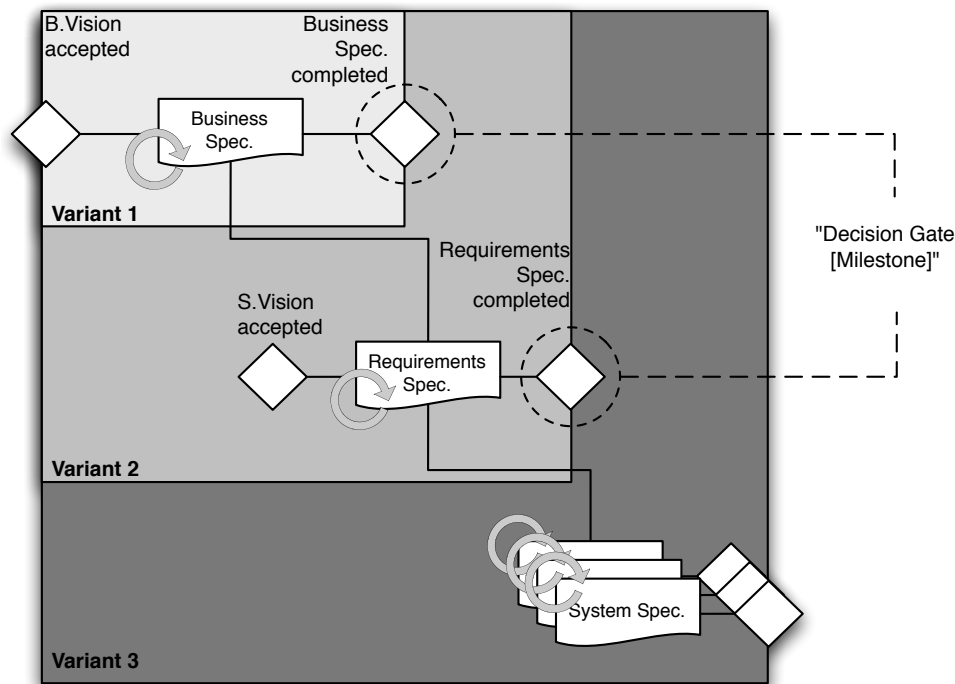


Figure 5.7: Variations in a project execution and decision gates from the perspective of the BISA reference model.

Each of the three coloured shapes, incorporating an artefact type and related milestones (see Sect. 4.8.3), represents a possible project variant extending each other.

³ Taking the artefact-based reference model for BISA, the objectives and the project scope are defined within the business vision.

5.3 Customisation at Project Level

We distinguish, in particular, the creation of the business specification and / or of the requirements specification and / or of the system specifications, each potentially considered by a particular project (see also example 5.3).

Example 5.3. *Variations in a Project Execution*

It is, for example, possible to exclusively create the business specification if considering a “consultancy project” (type) in which the project scope consists in a re-design of the business processes. It is also possible to create both the business and the requirements specification, e.g., if considering a multi-staged bidding procedure (or its assistance from the perspective of a contractor). Having created the requirements specification, a development project can be initiated. It is, however, also possible to consider all artefact types within one project.

How the milestones (indirectly) relate to each other depends on the project setting. If considering with a project the creation of all artefact types, the milestones can overlap, e.g., the system vision in the requirements specification can be created and accepted before the business specification is completed.

The definition of a frame contract for a particular project can be performed when reaching the first milestone of the artefact type on which the projects’ results rely. For example, if considering a development project, a frame contract can be defined on the basis of the system vision, which includes a brief description of the boundaries and the functionality of the system under consideration. When completing the last artefact type considered by a project, the project ends.

Finally, when reaching the milestone, which indicates when to complete the corresponding (agreed) artefact type, a decision has to be taken about whether to continue or to stop the project. In the following section, we thus use these milestones as *Decision Gates* at which we reflect on the current project situation.

5.3.2.3 Artefact-based Decision Support

In this section, we clarify the principles of artefact-based decision support needed for the customisation approach depicted in Fig. 5.5 on page 160. For this, we show what decisions can be taken by the use of the artefact-based reference model in contrast to if referring to activity-based approaches. Afterwards, we conclude with a discussion on the project repository itself and its interplay with decision support.

Decision Paths and Decision Gates

We see decision support as a means to show on what project characteristics to reflect and what consequences the taken decisions can have. For this, we give guidance on what parameters take effect on the creation of the artefacts, illustrated by the relation of concepts for project characterisation to the meta model for artefact orientation (see Fig. 5.6 on page 161).

The notion of decision support is two-fold. On the one hand, we can guide the dynamic content creation towards a balanced problem orientation by showing what project parameters impact what artefacts and, thus, on what project characteristics to reflect. On the other hand, when reaching a defined milestone (decision gate), we can reflect on whether the created artefacts are underspecified and what impact they have on further activities.

Figure 5.8 illustrates both resulting views. On top, we depict the relevant parts of the integrated BISA reference model. The middle of the figure illustrates the dynamic content creation. The bottom part of the figure illustrates the reflection at the decision gates.

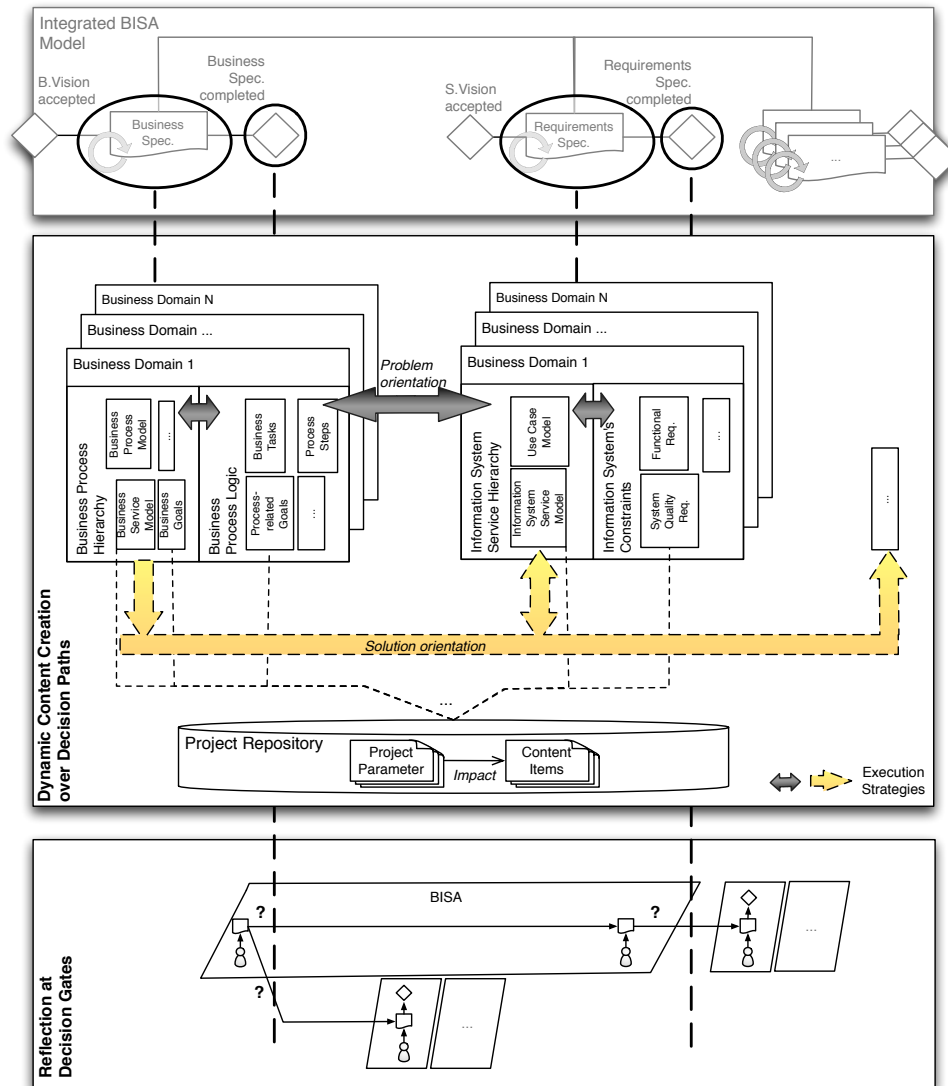


Figure 5.8: Artefact-based decision support: support for dynamic content creation (in the middle) and for reflection at decision gate (at the bottom).

Dynamic Content Creation. For the dynamic content creation, we make use of the project repository, which persists organisation-specific project parameters impacting the creation of selected artefacts. We use the project repository to guide the reflection on project characteristics for the content of each business domain and within this content for the content items at each level of abstraction (see also the process model in Sect. 4.8.2). The *Decision Path* itself, i.e., when to reflect on what parameter, arises from the content dependencies in the artefact model and the content items to be created next at each level of abstraction. The made decisions lead to a particular *Project-specific BISA Execution Strategy*, i.e., whether the performed

5.3 Customisation at Project Level

process is solution-oriented or not (as an outcome of potentially underspecified artefacts in response to particular project parameters).

Reflection at Decision Gate. At each of the decision gates, we are now able to reflect on potentially underspecified artefacts (see Sect. 4.2.3) as a response of the performed project-specific BISA execution strategy. The impacts on further development activities can objectively be estimated, because of the reference model being integrated into a development process model taking into account the interfaces in Sect. 4.7, respectively because of the dependencies of the artefacts to further artefacts that rely in their creation on related content items.

Project Repository and BISA Diary

As already shown, decision support is enabled by means of a project repository and by means of the BISA diary, which we both explain in the following.

Project Repository. A project repository persists project characteristics and their impact on the creation of artefacts (see Sect. 5.3.2.1). The persisted project characterisation then enables awareness during the creation of artefacts by assisting decision taking, i.e., by making the impacts of project parameters on the artefact types in the artefact model explicit. We refer to a project repository in particular within multi-project environments, where the project repository

1. is initially constructed once for an organisation on the basis of initially collected project parameters (see the analysis in Sect. 5.2.2.1),
2. is then used (after a training phase) in single project environments as an assistance to show on what project parameters to reflect for the creation of particular content items (and what decisions have been taken in other projects), and then
3. used after project completion to persist the own taken decisions via the BISA diary making potentially discovered new project parameters available for further projects.

In any case, we see the project repository as a means to exclusively assist decision taking by showing experienced impacts of project parameters on artefacts. This provides decision support instead of suggesting which decision to take (whether to create certain content items or whether to leave them underspecified).

BISA Diary. The BISA diary is used to document the decisions made in particular projects. This achieves a reproducible (BISA) execution and the persistence of these decisions in the project repository making them available to other projects.

As already shown in the overview depicted in Fig. 5.5 on page 160, we distinguish two customisation stages, which we take as orientation to organise the BISA diary. The diary is thereby organised by two major content items: The *Project Setting* and the *Project-specific BISA Execution Strategy*. The first serves to document the information that results from the initial project set-up, the second serves to document the information that results from the project-specific BISA execution strategy. The resulting structure of the BISA diary is illustrated in Fig. 5.9 and explained in the following. For the content of both items, we now take the meta model in Fig. 5.6 on page 161 as orientation.

The project setting includes project characteristics known from the beginning of the project when setting it up; for example, the *Project Background* describes properties of a project (the customer, the project scope, etc.). The *Process Frame* describes

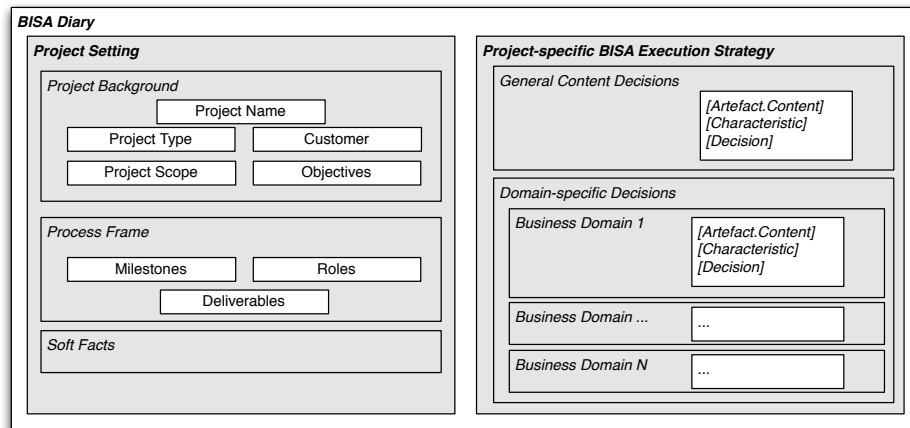


Figure 5.9: Structure of the BISA diary.

the outcome of the initial project set-up, i.e., the agreed deliverables (artefact types and their preliminary structure), the assigned roles, and the defined milestones. Inspired by the artefact *Project Diary* in the V-Modell XT, we use *Soft Facts* to describe further remarkable occurrences, which help to understand the overall project setting (like budget or experiences with the stakeholders).

The project-specific BISA execution strategy then documents the decisions taken during the dynamic content creation and gives a rationale for the content in the artefacts being created (or not) due to particular project parameters. For this, we distinguish general content decisions and ones that can be allocated to the content of a particular business domain.

Example 5.4. *(Business) Domain-specific Decisions versus General Decisions*

An example for a “General Content Decision” would be to remove certain requirements attributes in the concept model. An example for a “Domain-specific Decision” would be to document information system services instead of use cases for the business domain “Sales” due to the unavailability of user groups that would be necessary to approve concrete interaction scenarios with the system.

Each project characteristic and each of the made decisions (the actual impact on an artefact) is documented during customisation in corresponding sections that indicate to the customisation stage and the affected content items.

5.3.3 Approach for an Artefact-based Customisation

By now, we have discussed the principles of artefact-based customisation transferring related work in activity-based approaches. Based on the given foundation, we can subsequently describe how to perform the two customisation stages: the *Initial Project Set-Up* and the *Project-specific BISA Execution Strategy*.

5.3.3.1 Approach for Initial Project Set-Up

The initial project set-up describes the creation of exemplars of each of the sub-models of the artefact-based reference model.

5.3 Customisation at Project Level

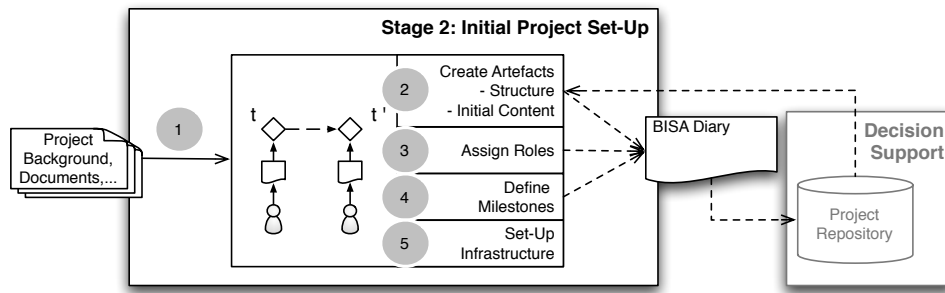


Figure 5.10: Approach for an initial project set-up.

In a first step, we elaborate the project background. In subsequent steps, we define the process frame, which considers the creation of the artefacts, the assignment of roles, the definition of milestones, and the set-up of the tool infrastructure. Figure 5.10 depicts the steps and in which of these steps we make use of the project repository, respectively of the BISA diary.

1: Elaborate Project Background. The elaboration of the project background considers the structuring of any kind of information about the project setting. Depending on the project setting, discussed in Sect. 5.3.2.2, background information can result from bidding documents or from foregoing projects in which initial specifications have been created.

Based on this information, the process frame can be prepared for subsequent steps (see the BISA diary), which, in turn, are performed in close collaboration with a customer, e.g., during a kick-off workshop.

2: Create Artefacts. The instantiation of the artefact-based reference model begins with the creation of the artefacts, on which all project participants have to agree. We distinguish between the creation of the artefact structure, i.e., the creation of the artefacts respecting the structure model, and the documentation of initial contents resulting from the project background, such as from initially given project requirements.

The artefacts are initially created due to the project type and the project scope. During this creation, it is possible to omit certain content items from the beginning and thereby to begin the project with underspecified artefacts.

Example 5.5. *Omitting Content Items*

An exemplary content item that could be omitted from the beginning of a project could be the “Business Demands Analysis”. An exemplary reason for removing this content item could be confidentiality, i.e., the unavailability of business internals necessary to calculate, e.g., business risks.

However, since every content item has its purpose and value, none of the content items are declared as optional or as mandatory, whereby the exclusion of content items must be performed in awareness of the effects on potentially necessary content items (arising from the content dependencies in the artefact model). The decision support for removing selected content items is additionally given by project

parameters that indicate the necessity of particular content items (the “Need for Action”).

When instantiating the artefact types with particular content items, these have to be

1. structured in the desired arrangement, for which we describe different possibilities in appendix B.1, and
2. named (where possible) according to customer-specific conventions.

The latter includes the identification of the relevant business domains, which serve to structure the content of the artefacts, but also to plan the subsequent workshops for elaborating the artefacts’ contents. Initially available contents, resulting from given background information (specification documents, bidding documents, etc.), can be classified into specific concept types and initially recorded within the created artefacts. These initially documented content items are then detailed during the project-specific BISA execution.

3: Assign Roles. Each of the created artefacts is assigned to a project participant that takes the responsibility for the artefact. Note that it is possible to assign same project participants to several roles (see example 5.6).

Example 5.6. *Assigning Roles*

One example for the assignment of project participants to the roles could be to assign “Mr. Sellers” to the business analyst and “Mr. Gonzalez” to the requirements engineer. Depending, e.g., on the temporal availability and the technical ability, it is also possible to assign “Mr. Sellers” to both roles, i.e., to the business analyst and to the requirements engineer and as such he then has the responsibility for both artefacts.

4: Define Milestones. After agreeing on the artefacts and the roles, the milestones are defined. Besides the definition of the milestones that define when to accept the vision documents and the ones which consider the completion of the artefacts (and which we see as decision gates, see Sect. 5.3.2.3), we can also plan further points in time for approvals, e.g., for workshops with stakeholders. These workshops can be structured according to the already known business domains, which serves to keep the amount of stakeholders to a minimum level and to be able to articulate clear goals in each of the workshops (see also the description of the iterative content creation in Sect. 4.8.2).

5: Set-Up Infrastructure. Although tooling is not in the primary scope of the thesis, the set-up of a tool infrastructure has to be considered during the initial project set-up. This includes the agreement on the used requirements management tools and on the CASE tools (see Sect. 2.2.5). The choice of the RM tool depends on aspects like the expected amount of requirements, the need of traceability and impact analyses during, e.g., change requests, or on given customer-side restrictions. Similar dependencies are given for the choice of CASE tools, which have to enable the creation of the artefacts w.r.t. the elements and relations defined in the concept model.

5.3.3.2 Approach for Project-Specific BISA Execution Strategy

Once the process frame is defined, the content of the artefacts is created within the defined set of structured artefacts and milestones. We showed in Sect. 5.3.2.3 that we are, due to the artefact-based philosophy, able to assist decision taking during this dynamic content creation in two ways.

On the one hand, we can assist the content creation itself by supporting awareness of project parameters impacting the possibility and necessity of creating specific content items. We showed that this assistance can be considered for the content of each of the identified business domains, in which the project characteristics may differ; for instance, user groups of business processes of one business domain could be available, while in another one, they are not. By transferring the principles of decision support, given by related work, to the artefact-based philosophy, we argued that the assistance then involves, same as during project set-up, the project repository. The project repository then characterises project execution by means of (arising) project parameters in relation to certain content items of the BISA reference model (see Sect. 5.3.2.1). Each of the made decisions is then documented in the second section of the BISA diary to achieve a reproducible process and, in particular, to make the made decisions available for other projects.

On the other hand, we can assist the reflection on the actually performed BISA execution strategy and potentially underspecified artefacts. This reflection can be performed at the milestones that indicate the agreed point in time when to complete the corresponding artefact type (see the decision gates in Fig. 5.7 on page 163).

Hence, we are able to systematically assist decision taking towards a balanced problem orientation achieving awareness of syntactically complete (or incomplete) artefacts. Figure 5.11 depicts the resulting approach, which we explain in the following.

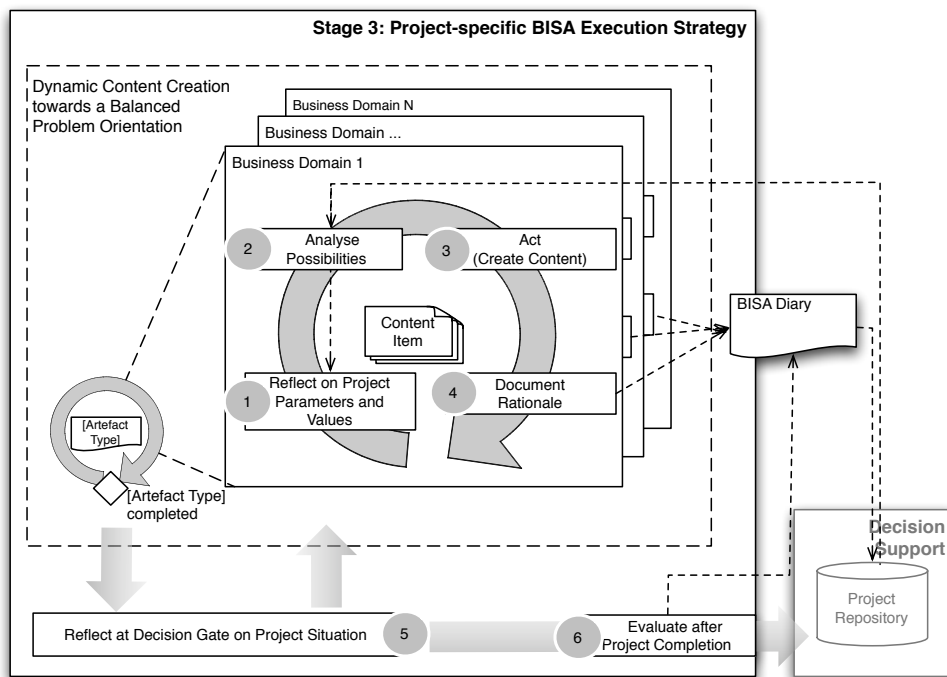


Figure 5.11: Approach for a project-specific BISA execution strategy.

1: Reflect on Project Parameters and Values. For each of the content items to be created (or completed), we need to reflect on the current project characteristic affecting this particular item. We take into account project parameters that hamper the creation of the content items (“Ability to Act”) and ones that argue for the creation (“Need for Action”). To assist this reflection, we also take into account the project characteristics persisted in the project repository by following the impact scales from the content items backwards to the project parameters and, thus, achieve awareness of the actual project characteristics.

Example 5.7. *Reflecting on Project Parameters*

An exemplary project characteristic that argues for the creation of a use case model could be a high degree of expected user interaction. At the same time, it can be possible that this creation is hampered by a weak access to business process specifications of a customer’s organisation and by the unavailability of user groups.

2: Analyse Possibilities. Due to the given project characteristics, we analyse the different possibilities in the content creation. In case there are, for example, project parameters that hamper the creation of certain content items and, at the same time, argue for the creation of others, we reflect on whether the other content items can be created (see example 5.8).

When analysing project parameters that potentially argue for leaving certain content items underspecified, directly related content items whose creation would be negatively affected, have to be taken into account by checking whether these can at least be created. This analysis is supported by the project repository the same way as described in example 5.8.

Example 5.8. *Analysing Possibilities of Content Creation*

If the creation of the use case model is hampered by the unavailability of user groups, this project parameter could argue for the creation of information system services, but also for the creation of the risk status report and additional obligations, used to compensate implied risks. We then use the project repository to check if there are project parameters that would negatively affect the ability of creating both the risk status report and the obligations, and check if these parameters also apply to the current project situation.

3: Act. After analysing given possibilities for creating the content items or alternative ones, these are created according to the concept model.

4: Document Rationale. Having taken the decision about how to continue for particular content items of a business domain, the rationale is documented in the BISA diary in corresponding section.

5: Reflect at Decision Gate on Project Situation. When completing the content of each business domain in an artefact to its possible and necessary extent, we reflect on the overall reached project situation at the decision gate. We analyse what content items are left underspecified as a consequence of the performed BISA

5.3 Customisation at Project Level

execution strategy. For each of the underspecified content items, we check the associated content items. For this purpose, we defined in Sect. 4.7 to what general activities certain content items relate.

Example 5.9. *Reflecting at Decision Gate*

If system quality requirements of a particular business domain remain underspecified (e.g., due to a missing operative background), are they compensated with normative references? Do these normative references provide enough information for (technical) design activities?

We take into account underspecified content items with their potential consequences on other artefacts and the documented project parameters that give the rationale for the content items. Based on this information, we can estimate risks and take the decision about whether to continue the project or not.

However, although we are able to support the reflection on the overall project situation, the final decision itself cannot be formalised as it also depends on strategic issues and on human nature.

6: Evaluate after Project Completion. Finally, after project completion, the decisions that have been documented in the BISA diary have to be persisted in the project repository. To avoid taking over questionable decisions, these have to be evaluated. This evaluation can only be done to a certain level. One possibility is to estimate if certain decisions have lead to additional change requests that did not result in projects with similar characteristics.

However, the evaluation of the project parameters and the decision about whether to extend the project repository with project parameters that have not been taken into account yet, must remain in the end to project lead and his individual estimation.

CHAPTER 6

Case Studies

In this thesis, we contribute an artefact-based customisation approach for requirements engineering. This approach aims at supporting an integration of our artefact-based reference model into a development process model and the customisation of the integrated reference model at project-level in order to tackle current shortcomings in available (activity-based) approaches.

In this chapter, we evaluate our contributions with two case studies, which we organise according to the introduced customisation stages. As a first step, we provide a case study considering a process integration of our artefact-based reference model into a development process model (Sect. 6.2). As a second step, we provide a case study on customisation of the integrated reference model at project level in an industrial setting (Sect. 6.3). In Sect. 6.4, we discuss our results, their limitations, and their implications. The content of this chapter is based, in part, on our contribution in [MFLPW11].

At the end of this chapter, the reader will know the advantages and limitations of our contributions and whether we finally achieved our objectives.

Contents

6.1	Case Studies Overview	174
6.2	Case Study 1: Process Integration	175
6.3	Case Study 2: Customisation at Project Level	189
6.4	Discussion and Conclusion	201

6.1 Case Studies Overview

We perform two case studies in order to investigate the advantages and limitations of our contributions with respect to the objectives defined in Sect. 1.2. These objectives consist of establishing an artefact-based reference model for RE and offering guidance for

1. a process integration of the reference model into a development process model.
2. a customisation of the reference model at project level. This means to assist the flexible content creation of the artefacts in response to individual project situations that affect the creation of those artefacts.

For this, we perform two case studies that evaluate our contributions. Each of the case studies considers the evaluation of a particular customisation stage in a qualitative manner. Figure 6.1 illustrates an overview of both case studies and their interrelation.

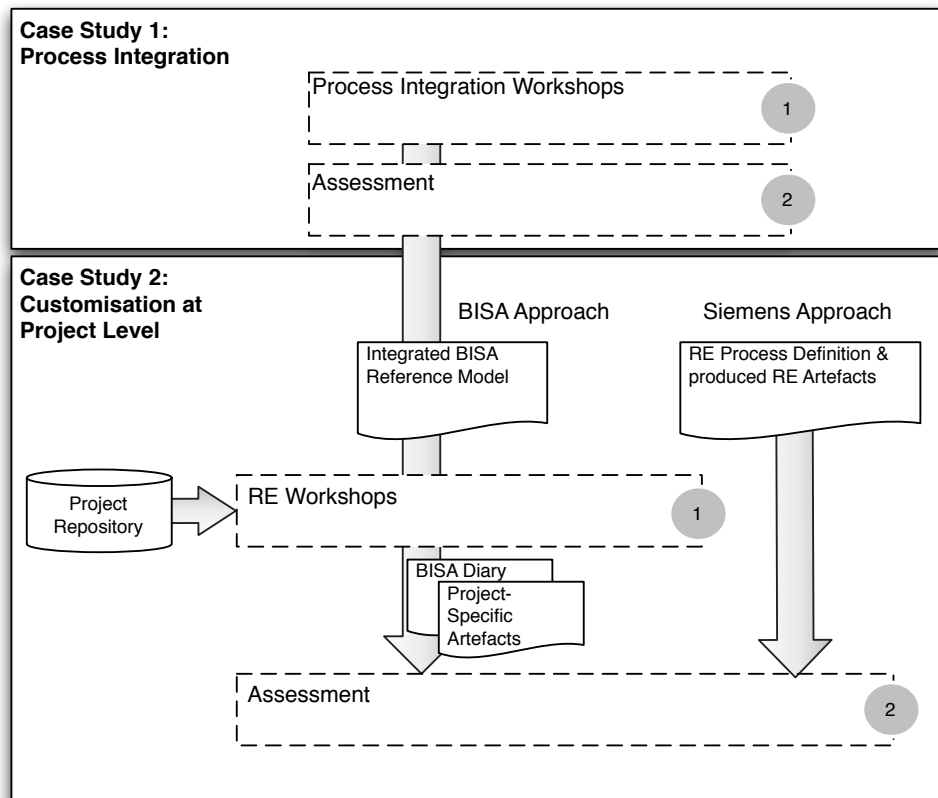


Figure 6.1: Overview of case studies.

The first case study (Sect. 6.2) analyses the application of our contributions at organisational level. The second case study (Sect. 6.3) analyses the application of our contributions at project level in an industrial setting.

The studies are both performed as an action research in which the author directly participates in the procedure having not an exclusively observational role. Both case studies are briefly introduced in the following.

Case Study 1: Process Integration. In the first case study, we analyse the process integration of our artefact-based approach by performing an action research study. We perform a process integration of our artefact-based reference model into the V-Modell XT by following the procedure presented in Sect. 5.2. The goal of the study is to assess the applicability of the process integration procedure. The motivation for choosing the V-Modell XT lies in achieving an improvement of the development process model by enriching its content with a domain-specific requirements engineering approach, which yet is missing. After completing the process integration, we generate an overall process documentation, which we assess according to a set of criteria. The generated artefact templates (part of the process documentation) are then used at project level in the second case study.

Case Study 2: Customisation at Project Level. In the second case study, we apply our artefact-based approach in a real development project to analyse advantages and limitations with respect to the activity-based RE approach previously used in the same industrial context. Since our artefact-based reference model has been, in parts, developed in co-operation with Capgemini TS (see Sect. 3.3.3), we change the setting to another company. The new industrial context is hosted by a development department of the company Siemens.

In the case study, we apply our approach in a series of RE workshops by following the customisation procedure described in Sect. 5.3. We prepare the process and perform a preliminary customisation of our reference model taking into account initial documents provided by Siemens. In addition, we prepare the project repository, necessary for customisation, by referring to a set of project parameters gained from our previously performed field study at Capgemini TS [MFWL⁺12]. This transfer of project parameters from one industrial environment to the next shall demonstrate the validity of not only the overall approach, but also of the parameters themselves. After performing our approach, we assess it in a direct (benchmark) comparison to the previously used process according to a set of criteria.

6.2 Case Study 1: Process Integration

The goal of this study is to evaluate the process integration capabilities of our contributions: the artefact-based reference model to be integrated into a development process model (Chp. 4) and the corresponding customisation procedure (Chp. 5.2). In this case study, we take the perspective of a process engineer, who customises the development process model for a company-wide use.

In the following, we introduce the research objectives and the context of the study. In Sect. 6.2.2, we describe the study design, which we organise according to the guideline of Runeson et al. [RH09]. The results of the study are presented in Sect. 6.2.3 and discussed in Sect. 6.4.

6.2.1 Research Objective and Context

Our objective is to evaluate the applicability of our approach when integrating the BISA reference model into a development process model.

One possible study design would include a benchmark comparison of our approach with existing ones. When following such a design, we would perform the

6.2 Case Study 1: Process Integration

process integration of our BISA reference model in direct comparison with the integration of an activity-based approach and, thus, follow the research objective to investigate possible improvements in the process integration.

Such a comparative study is, however, not possible. Necessary cases that could serve as a reference for an activity-based process integration are, to the best of our knowledge, not available:

1. Available activity-based development process meta models like SPEM (see [OMG08] and Sect. 2.5.3.1), do not contain any guidance for customisation or process integration that could be assessed (see also the related work discussion in Sect. 2.6.2.1).
2. Those meta models, which include guidelines for their application, as found in the area of organisational method engineering, focus on the (from scratch) construction of domain-specific development process models, but not on the integration of selected parts into existing models. An exemplary meta model-based approach, which relies on this area of method engineering, is given with *MetaME* [ES10].
3. Those activity-based approaches, which focus on integration aspects into given development process models, only take into account the integration of single description techniques or single methods. They disregard the associations of the methods to further parts of the development process model and, thus, they do not sufficiently cover the needs for performing a comprehensive process integration (see also the discussion on customisation principles in Sect. 2.6.1.1).
4. Available domain-specific approaches (e.g., for RE), which rely on the activity-based philosophy so that we could integrate these approaches into a development process model by ourselves, do not cover all sub-models of a development process model (methods, milestones, roles, artefacts, etc.). Hence, we cannot reproduce the process integration even without having guidance for this integration.

Despite the missing possibilities to reproduce an activity-based process integration, we must see such a reproduction, in general, as critical, because (without guidance) it would be undermined by subjective opinions of the study participants. Since there does not exist any empirically sound experience reports on such an integration that could be used for a comparison, we exclude a comparative study design at all.

Therefore, we aim at performing the process integration by following the steps in our customisation approach and by directly assessing the outcome of the integration. We formulate the objective according to the goal definition template [WRH⁺02] in Tab. 6.1.

Table 6.1: Research objective and context.

Analyse	the process integration of the artefact-based reference model
for the purpose of	evaluation
with respect to their	applicability
from the point of view of the	process engineer (process author)
in the context of	a meta model-based development process model

6.2.2 Case Study Design

In the following, we describe the case study design. As a first step, we define the research questions of the study, describe then the case and subject selection, and the analysis procedure. We conclude with a discussion on the validity procedure.

6.2.2.1 Research Questions

The goal is to investigate the applicability when process-integrating our artefact-based reference model into a development process model following our proposed approach. Since it is not enough to simply estimate whether an approach is applicable or not, we formulate two research questions. These consider both the usability of the actually performed process integration and the quality of the produced results.

RQ 1. *Is the process integration approach usable to perform a systematic process integration of the artefact-based (BISA) reference model?*

We want to know whether the procedure proposed in Sect. 5.2.2 offers enough guidance to perform a systematic process integration of our artefact-based reference model.

RQ 2. *Does the performed process integration lead to usable results?*

Once we performed the process integration, we still have no detailed insights into its error-proneness and, thus, we need to know of what quality the produced results are. For this, we analyse whether the overall generated process documentation is usable as a self-contained development process model, which suitably covers the BISA-specific contents.

6.2.2.2 Case and Subject Selection

In the following, we describe the case and the subject selection.

Case Selection. For integrating our artefact-based reference model into a development process model, we need to choose a development process model that fulfils two prerequisites. First, it must comprehensively cover with its activities, artefacts, and roles an overall development process. Second, it must enable an analysis of its structural properties and an interpretation of the underlying concepts, i.e. we must be able to unambiguously map the elements and relations defined by our meta model (Chp. 3) on the elements and relations given by the envisioned development process (meta) model. As a consequence, the chosen development process model must be based on a meta model and underlie a same or similar artefact-based philosophy as given in our approach (see also Sect. 5.2.2.1 describing the prerequisites for a process integration).

Subject Selection. We define two groups of participants as study subjects:

1. *Process Engineer:* We assign a process engineer who is familiar with the chosen development process model and with our approach. He is responsible for performing the actual process integration and assessing the results.
2. *External Reviewer:* We additionally assign an external reviewer to objectively assess the produced process documentation. He must not be involved into

6.2 Case Study 1: Process Integration

the actual integration process. He needs, however, knowledge about the chosen development process model in order to objectively (and effectively) assess the produced results.

6.2.2.3 Data Collection Procedures

We collect the data for our case study in two steps. First, the process engineer performs (supported by the author of the thesis) the process integration in a series of workshops following the steps proposed by our approach. After having completed the process integration, we perform the assessment of our approach. The process engineer assesses the actually performed approach and the resulting process documentation. We additionally call in the external reviewer to exclusively assess the process documentation.

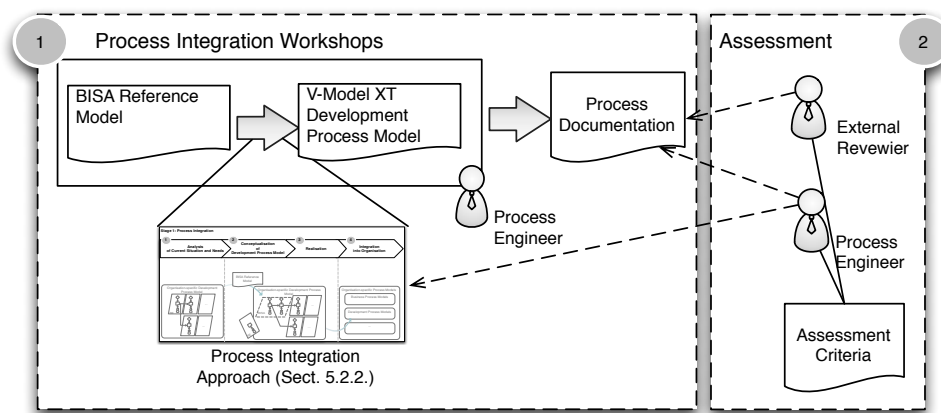


Figure 6.2: Overview of data collection procedure.

Figure 6.2 illustrates the data collection procedure, which we explain in the following.

Process Integration Workshops

We perform the process integration in a series of workshops. In each workshop, the process engineer performs, supported by the author of the thesis, a particular step of the approach presented in Sect. 5.2.2:

1. Analysis of Current Situation and Needs
2. Conceptualisation of Development Process Model
3. Realisation

A restriction of the case study is that we intentionally do not take into account a real company-specific environment and exclusively focus on the technical aspects of the process integration. Therefore, we omit the fourth step in the process integration approach presented in Sect. 5.2.2 (“Integration into Organisation”). In the following, we introduce the three process integration workshops.

Analysis of Current Situation and Needs. At a kick-off workshop, the author of the thesis presents the BISA reference model, the underlying meta model, and the process integration approach. Afterwards, we perform an analysis of the development process model. For this, we take into account the meta model of the development process model to establish a terminological and structural mapping

between the concepts used in our meta model and the ones used in the envisioned development process meta model. In addition, we analyse the development process model with respect to initial development phases to identify typical artefacts, activities, milestones, and roles that are important to requirements engineering.

Conceptualisation of Development Process Model. We conceptualise the integration in a sketch and define typical (project execution) scenarios in which the RE-relevant elements in the development process model are applied. We customise (create / replace / modify) these identified elements according to the BISA reference model. We begin with the modification of the artefacts, define the corresponding roles, the milestones, and conclude with the definition of the activities.

Realisation. In the last workshop, we realise the drafted concepts. If available, we make use of development process modelling tools and use these tools to export the process documentation.

Assessment

After completing the process integration workshops, we perform the assessment. For this, we develop a questionnaire, which we propose to the process engineer and to the external reviewer as a preparation for the assessment. The actual assessment is performed as a structured interview, in which the study participants fill in the questions given in the questionnaire. This interview is performed separately with the external reviewer and the process engineer (both in isolation).

The questionnaire is organised according to a set of chosen criteria, which we use to answer our research questions. For each criteria, we define an open and a closed question. The closed question demands for the agreement to a given positivistic statement, to be answered on a Likert-scale from *1 = I strongly disagree* to *8 = I strongly agree*. We deliberately choose an 8-point scale to avoid that the interviewees check the middle. The open questions are used to express their expert opinion as free text. The answers also serve as a rationale to reproduce the chosen agreement to the statement of the closed questions. The condensed questionnaire with the statements of the closed questions is shown in Tab. 6.2.

We define seven criteria to answer the first research question (the usability of the process integration approach) and seven criteria to answer the second research question (the quality of the produced process documentation). The corresponding questions for the first research question are exclusively answered by the process engineer. The questions for the second research question are answered by both the process engineer and the external reviewer.

To answer RQ 1, we ask as a first step for the overall impression the process engineer got when applying the approach. We are, for example, interested in the ease of use. Furthermore, we are interested in the overall experienced structuredness, which we see as given if the approach offers enough guidance to consciously cope with the situations to be faced during process integration. We are, however, not only interested in a rating of the overall approach, but also in a rating of selected steps. For this, we distinguish between the rating of the unambiguity of the analysis procedure and of the procedure for performing the conceptualisation and the construction. This way, we can track down possible limitations of the approach to single steps, e.g., regarding the interpretation and mapping of the elements in both meta models during analysis. Further criteria are, e.g., the sustainability of the approach, which we see as an important criteria due to the general criticality of a process integration (which should be performed only once).

6.2 Case Study 1: Process Integration

Table 6.2: Questionnaire for the assessment (condensed closed questions).

	Criteria	Statement
RQ 1	Ease of Use	The overall approach is clear and understandable.
	Structuredness	The overall approach offers structured guidance.
	Unambiguity of Analysis	The approach offers guidance to unambiguously perform the analysis.
	Unambiguity of Conceptualisation & Construction	The approach offers guidance to unambiguously perform the conceptualisation and construction procedure.
	Minimality	The approach describes no unnecessary guidance.
	Sustainability	The approach is not error-prone.
	Productivity	The perceived (overall) productivity was high.
RQ 2	Syn. Completeness of BISA	The BISA reference model could be completely represented in the process model.
	Syn. Completeness of Dev. Process Model	No elements or relations in the development process model are lost.
	Compatibility of Integrated Elements	The integrated BISA elements are compatible with the depending ones.
	Information Loss in Artefact Dependencies	All dependencies between the BISA artefacts and further artefacts are covered.
	Ease of Perception	The integrated BISA reference model is understandable and communicable.
	Unambiguity	The content of the integrated BISA reference model is unambiguous.
	Content Improvement of Dev. Process Model	The quality of the process documentation improved after the process integration.

We answer RQ 2 by assessing the quality of the produced process documentation. We ask, for example, for the (syntactic) completeness of the produced documentation with respect to the BISA reference model and to the previously contained (RE-specific) elements. We are also interested in the compatibility of the integrated elements and possible information loss in the artefact dependencies. With compatibility, we consider whether the integrated elements offer enough information to create the elements interconnected via content dependencies. Regarding the information loss, we consider whether all associations between the artefacts (BISA-specific and further ones) proposed by our approach and / or by the development process model are captured. For example, we investigate whether we captured the associations between testing-specific artefacts and contents in the requirements specification. We further assess the actually integrated content for, e.g., the ease of perception. Finally, we conclude with analysing whether we could achieve after the process integration an overall improvement in the development process model.

6.2.2.4 Analysis Procedures

Due to the chosen study design, statistical hypothesis testing is not applicable. Hence, we present the results of the closed questions as a chart and analyse the answers to the open questions qualitatively to further explain the answers of the closed questions.

6.2.2.5 Validity Procedures

To decrease the failure rate in the performed process integration workshops, we strictly organise them according to the steps in our approach.

Regarding the assessment, we increase the internal validity by involving an external reviewer into the assessment. To mitigate the threat of a bias toward our approach by this external reviewer, he is not involved in the study prior to the actual assessment and exclusively rates the produced process documentation. He is furthermore not allowed to access further documentation to ensure that his judgement relies only on the produced documentation.

Finally, we use cross examination by asking both open and closed questions. The open questions give both the external reviewer and the process engineer the possibility to freely express their expert opinion. The closed questions force each of them to agree on one statement.

6.2.3 Case Study Results

In the following, we describe the chosen cases and the involved subjects. We conclude the section with a description of the analysis results (of the assessment), which we structure according to our research questions.

6.2.3.1 Case Description

We conduct the case study by integrating the BISA reference model into the *V-Modell XT*. We already introduced the basic concepts of the *V-Modell XT* in the fundamentals in Sect. 2.5.3.2. For the process integration, we rely on

- the *V-Modell XT Basic-EN, Version 1.1*, and
- the *V-Modell XT Meta Model, Version 1.3*.

To conduct the prototypical realisation of the process integration, we use two tools: the *V-Modell Editor, Version 3.3.8*¹ and the *PDE-Editor 0.9.5 for V-Modell XT 1.3* (Process Development Environment Editor)². The *V-Modell editor* is a tool that supports the modification of the *V-Modell XT* (persisted as XML) and the export of the process documentation, e.g., as a PDF document. The *PDE editor* offers similar functionality. It lacks, however, for an export function, but supports, in turn, a runtime check for conformance to the meta model version 1.3. In addition to those development process modelling tools, we use the *V-Modell XT Project-Assistant, Version 1.3.5*³. The project-assistant serves to initiate a project on the basis of a concrete *V-Modell XT*, i.e., to perform the tailoring and generate exemplars of the artefacts and a project plan. We use this tool to test whether we successfully integrated the BISA reference model including the corresponding tailoring profiles (see the subsequent sections).

In the following, we describe the case, organised by the single steps of the process integration.

Analysis of Current Situation and Needs. We first analyse both the *V-Modell XT* meta model (see the fundamentals in Sect. 2.5.3.2) and the meta model for artefact orientation (defined in Sect. 3.3). Both meta models rely on a similar artefact-based philosophy, i.e., the process is defined on the basis of chosen artefacts and

¹ Available at: <http://sourceforge.net/projects/fouever/files/>

² Available at: <http://pde.codeplex.com/>

³ Available at: <http://sourceforge.net/projects/fouever/files/>

6.2 Case Study 1: Process Integration

those process elements (activities, milestones, roles), which are coupled to those artefacts. We establish a mapping between the basic concepts of both meta models as shown in Tab. 6.3.

Table 6.3: Mapping of elements in the meta models (condensed).

Meta Model for Artefact Orientation	V-Modell XT Meta Model (1.3)
Phase	Discipline
Activity	Activity
Task, Method	Method Reference
Artefact Type	Product Type
Content Item	Topic
Concept Type	N/A
Milestone	Decision Gate
Content Dependencies	Creational Dependencies & Content-related Dependencies

In scope of the mapping are the concepts used to define artefacts. The V-Modell XT defines an artefact type as a *Product Type*, which can be decomposed into *Topics*. It offers, however, no means to express the content of the content item (via “concept types”), except for an informal description, e.g., of the purpose of a product type. We thereby make use of the informal description to describe the concept model and additionally define the content dependencies of the BISA model by means of directed *Creational Dependencies* and undirected *Content-related Dependencies*. The first dependency type describes why a particular product type has been created, the latter describes that the contents of the interconnected product types relies on each other.

Once we understood the interrelations of both meta models, we analyse the actual V-Modell XT in order to identify the relevant excerpt into which to integrate the BISA model. As described in the fundamentals Sect. 2.6.4.2, the V-Modell XT offers several tailoring profiles, which group a set of logically related process elements according to project types, project type variants, and further project variables. Within those profiles, the following are of interest, since these consider requirements engineering and related product types:

1. Project type *System Development Project (Customer)* with the project type variant *Project (Customer) with one Supplier*.
2. Project type *System Development Project (Customer / Supplier)* with the project type variant *Project (Customer/Supplier) including SW-Development, Enhancement & Migration*.

Both profiles consider the beginning of a development project with the creation of a *Project Proposal*. This project proposal is used in a potentially initiated follow-up project to create the *Requirements Specification* and further product types, which are of interest for requirements engineering, e.g., the *User Task Analysis* that corresponds to a use case model.

Conceptualisation of Development Process Model. We conceptualise the process integration with a whiteboard sketch and focus, as a first step, on the product types (see Fig. 6.3). We distinguish with corresponding annotations product types that are created from scratch, ones that are taken over from the V-Modell XT, and ones that are replaced by the content of the BISA model.

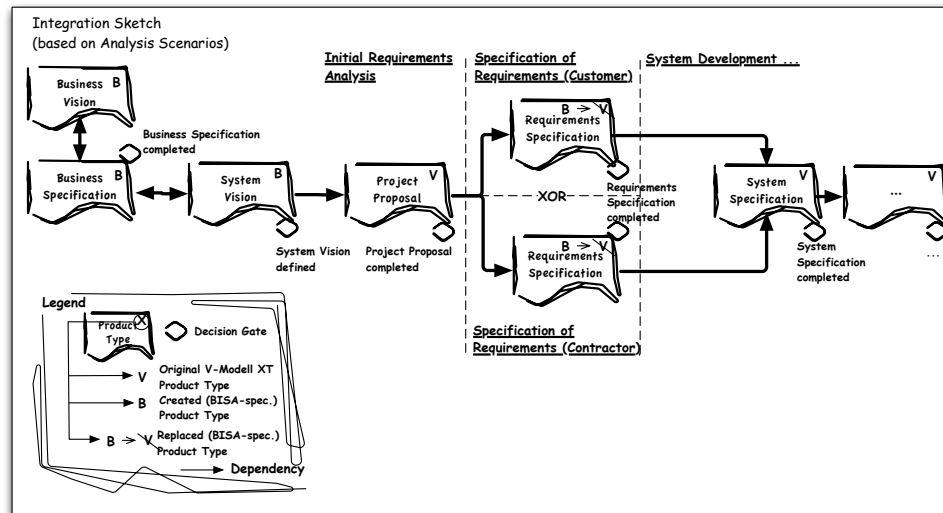


Figure 6.3: Sketch of the process integration with particular focus on artefacts (simplified).

We begin the integration by defining the (BISA-specific) product types and sketch corresponding process modules that contain those product types. The first process module to be considered is the “Initial Requirements Analysis”. This process module contains the project proposal (describing initial ideas of a system to be developed) and extends this product type with the system vision, the business specification, and the business vision. We define further process modules to contain the requirements specification (except for the system vision), which replaces the original requirements specification of the V-Modell XT.

Having integrated the product types, we conceptualise according to the process integration approach of Sect. 5.2.2

1. the dependencies between the (BISA-specific) product types and further V-Modell-specific ones taking into account the interfaces for the process integration of Sect. 4.7.
2. the corresponding roles; we create the business analyst from scratch and extend the existing requirements engineer according to our role model in Sect. 4.9.
3. the milestones for the corresponding product types.
4. the activities and the tasks.

Realisation. In order to facilitate the planned assessment, we focus during the realisation on the first process module sketched in Fig. 6.3. In addition, we reduce the content of the V-Modell XT with the V-Modell editor by selecting only those project types, which we see as necessary (see the analysis workshop). Furthermore, we remove un-referred elements with the PDE editor, e.g., the glossary or the method references. Otherwise, the resulting process documentation to be assessed would contain approximately 900 pages.

To realise the conceptualised process integration, we follow a top-down approach, since we first need to create the process-specific containers for the product types (see Fig. 6.4). We first define the discipline “Business Information Systems Analysis”, then the process module “Initial Requirements Analysis”, and finally the corresponding product types, each with a corresponding activity that gives guidelines for the product’s creation.

6.2 Case Study 1: Process Integration

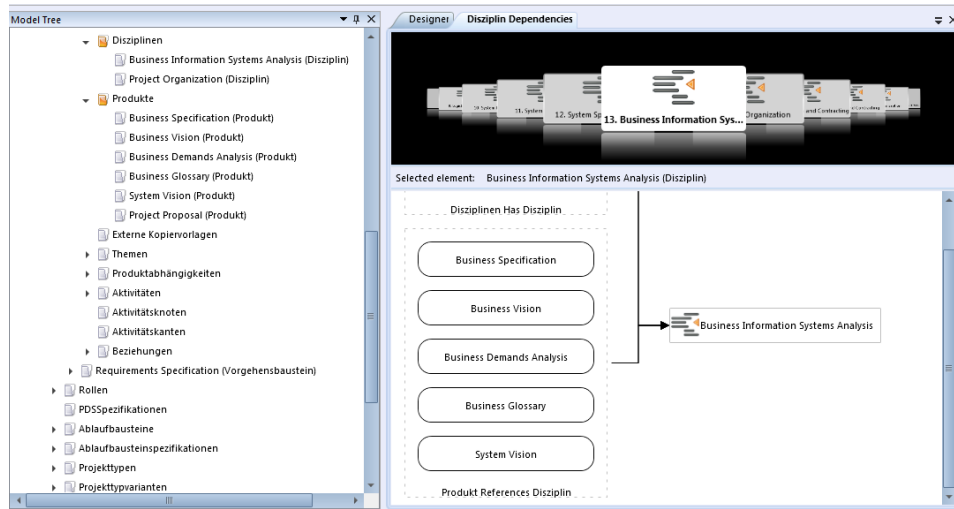


Figure 6.4: PDE screenshot: creation of disciplines and corresponding product types.

As described in the fundamentals, the V-Modell XT defines several project types necessary for customisation. To integrate the BISA model into this customisation approach, we make use of procedure modules. A procedure model is used to define a concrete work flow on the basis of decision gates. These modules are then computed during customisation to a concrete project execution strategy. Thus, to integrate the BISA model, we create a tailoring profile

1. by defining a project type *Project Study* and
2. by defining corresponding procedure modules, which we use to define project execution strategies.

The designed project execution strategy is illustrated in Fig. 6.5 and shows the design of the procedures in which to potentially reach which milestone.

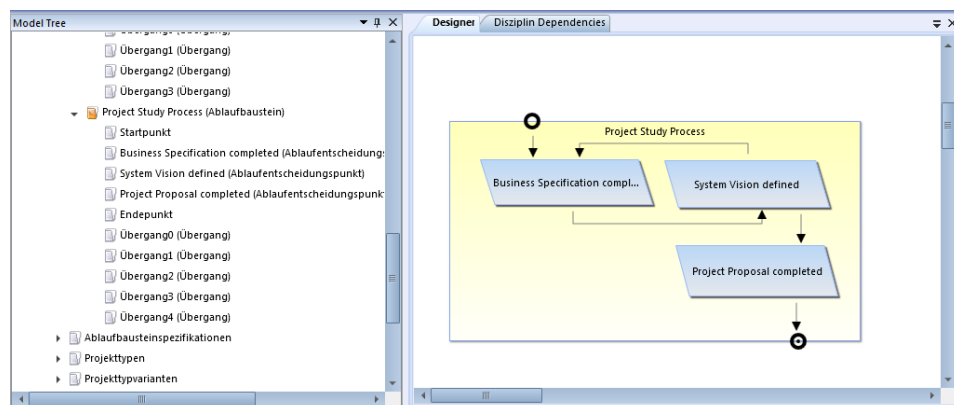


Figure 6.5: PDE screenshot: creation of procedure modules for project execution strategies.

After testing the tailoring profile with the V-Modell XT Project-Assistant, we continue the integration by

1. defining each product type in detail (with the necessary topic definitions and content descriptions),
2. defining the roles,

3. establishing the associations between the product types and the process elements (milestones, activities, roles), and by
4. establishing the associations between the product types.

The last step considers the establishment of the associations between BISA-specific product types and V-Modell XT-specific ones. Table 6.4 summarises the established associations. For reasons of complexity, we omit in this summary the associations within the BISA-specific products (e.g., the association between the business specification and the system vision).

Table 6.4: Associations between BISA-specific product types and V-Modell XT-specific ones.

BISA-specific Products	V-Modell XT Products
Business Specification	User Task Analysis
Business Vision	Project Manual
Business Demands Analysis	Life Cycle Cost Management
	Risk List
System Vision	Project Proposal
	Requirements Specification
	User Task Analysis
	Overall System Specification
	Contract
	Project Proposal
	Problem and Change Evaluation
	Estimation

Finally, we complete the process integration by exporting the resulting process documentation as a PDF, which we use in the assessment workshop.

6.2.3.2 Subject Description

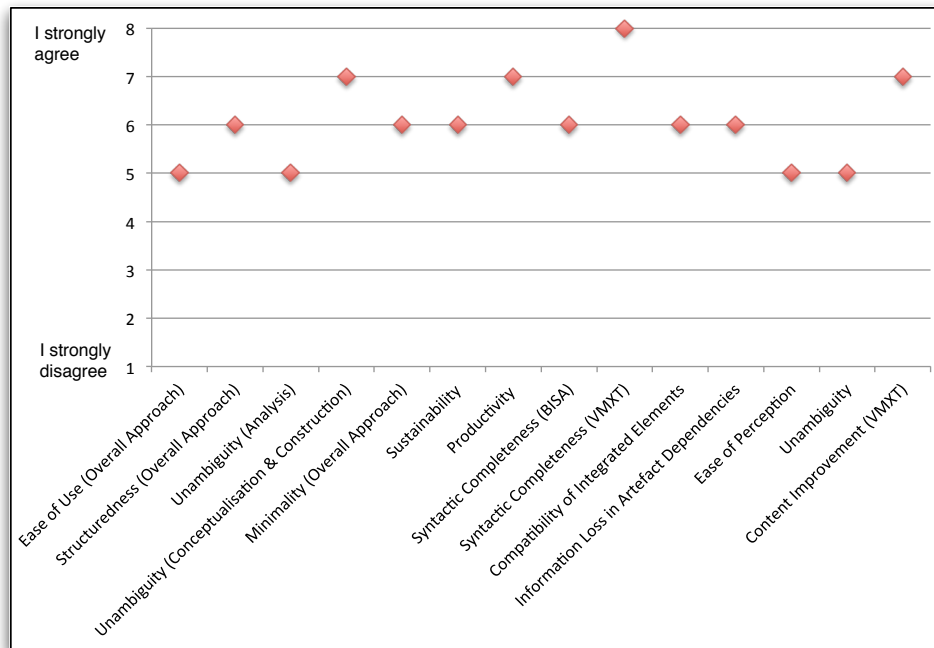
As introduced in the subject selection, we distinguish two study participants: The *process engineer* is responsible for performing the process integration (supported by the author of the thesis) and assesses afterwards the approach with the corresponding process documentation. The *external reviewer* exclusively assesses the process documentation without insights into the actually performed process integration.

The role of the process engineer is taken by Marco Kuhrmann, a researcher of the Technische Universität München. He participated in the development and the ongoing maintenance of the V-Modell XT and also has insights into the BISA reference model. As external reviewer acts Manuel Then. He also has detailed insights into the V-Modell XT and the corresponding tool set.

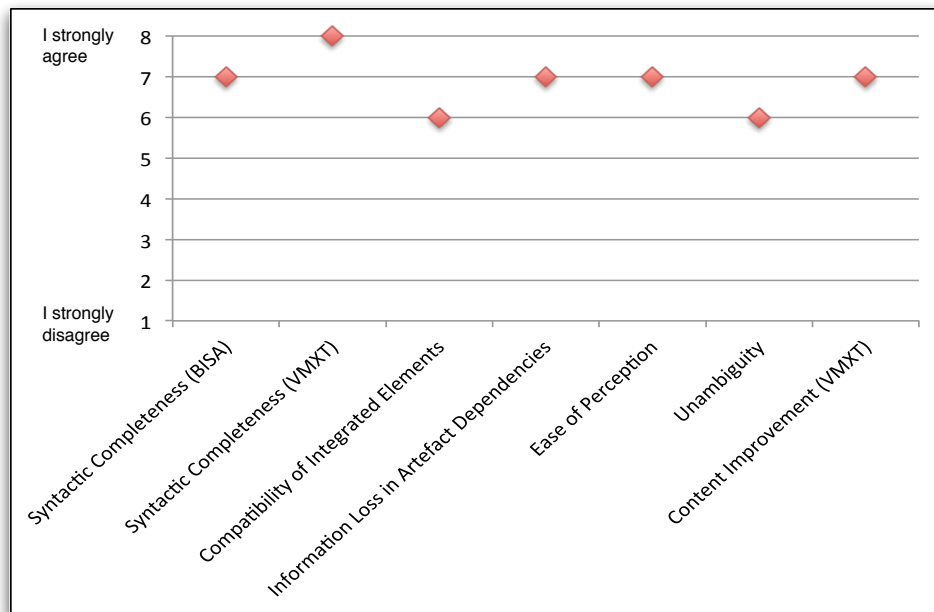
6.2.3.3 Analysis Results

Figure 6.6 illustrates the results of the assessment, depicting the ratings of the closed questions as charts. The upper chart illustrates the ratings given by the process engineer, the lower chart illustrates the ratings given by the external reviewer. In the following, we describe the results taking into account the answers given in the open questions.

6.2 Case Study 1: Process Integration



(a) Internal Review



(b) External Review

Figure 6.6: Analysis results shown as radar plots.

Usability of Process Integration Approach (RQ 1). The usability of the process integration approach was exclusively assessed by the process engineer. While the ratings vary for each of the assessment criteria, there is an overall trend to a positive rating with 5 points as the lowest agreement made for a given closed question.

Regarding the *ease of use of the overall approach*, he judged the approach to cover a sufficiently complete spectrum of process integration details. However, he also

stated the approach to be complex, since it relies on complex models (meta model and BISA reference model), whereby he also needed the direct support of the author of the thesis.

He also judged the *structuredness of the overall approach* to be given, but noticed the approach to be defined in a generic manner so that it covers a broad spectrum of development process (meta) models into which to integrate the BISA reference model. He stated the explicit need of customising the integration approach itself according to the specialities of different development process models (like the V-Modell XT).

This need is also reflected in the rating of the *unambiguity of the analysis procedure*, since the analysis of the meta models and the reference models relied, to some extent, on the domain knowledge given during the integration. The actual *unambiguity of the conceptualisation and construction procedure*, in turn, was stated as high, while based on the assumption that the results of the analysis procedure were valid.

The need of specialising the process integration approach according to different development process (meta) models takes similar effects on the rated *minimality of the overall approach*. The minimality of the approach suffers, although being generically described, from the actual trend to cover as many development process models as possible.

He judged the *sustainability of the performed process integration* to be given, but stated that the error-proneness also depends on the goals of the process integration and the used (tool) infrastructure. He judged that the fact of using a standardised development process model with the currently available tools directly supported the sustainability. He doubted, however, that such a sustainability would be given if referring to, e.g., the Software Engineering – Metamodel for Development Methodologies (SEMDM). In his opinion, such an integration would exclusively have an academic value due to the missing (tool) infrastructure.

Finally, the process engineer rated the *productivity of the performed approach* as high, because we needed three workshops (each of one day) to generate the overall process documentation. His judgment also relied on his experiences with previously performed modifications of the V-Modell XT.

Usability of generated Process Documentation (RQ 2). The usability of the generated process documentation is assessed by both the process engineer and the external reviewer. Both ratings vary, but show same trends.

The *syntactic completeness of the BISA reference model* was rated as high, since each element could be integrated with respect to the performed interpretation of both (meta) models during the analysis. However, this interpretation was acknowledged to lead to a certain information loss, since the missing possibilities to define a concept model for each artefact type (product type) had to be compensated by informal content descriptions and content-related dependencies. The external reviewer additionally stated that the content-related dependencies were, in parts, hardly comprehensible due to the informal content descriptions. The *syntactic completeness of the development process model* was rated, in turn, with the highest grade, because the process integration was even re-using existing elements in the V-Modell XT, e.g., the project proposal, and furthermore adding BISA-specific product types.

Since we reused some elements in the V-Modell XT, the participants also rated that the *compatibility of the integrated elements* could have suffered. In particular, they stated that the compatibility seems to be generally given. However, it could not be proven whether the resulting process documentation was still completely free of

6.2 Case Study 1: Process Integration

redundancies and / or contradictory assertions, whereby an additional case study would be necessary at project level.

Regarding the *information loss in the artefact dependencies*, they stated that all the dependencies between the BISA artefacts and the V-Modell XT product types could be expressed. The process engineer stated that this also resulted from the dependencies generically proposed by the BISA reference model (see Sect. 4.7). The external reviewer, in turn, stated that not all dependencies could be formally checked within the review of the process documentation, since the content-related product dependencies were established between product types of different process modules. Hence, his rating relied, to some extent, on assumptions and his existing knowledge of the V-Modell XT.

The participants agreed on the *ease of perception*, but judged that this also depends on the qualification of the team members who would apply the corresponding part of the process documentation. According to the process engineer, this criterion is not seen as an important measure of success, since the details in the contents have to be communicated at project level rather than at organisational level. The external reviewer, however, judged the contents to be easily communicable.

Regarding the rating of the *unambiguity*, the external reviewer judged that the unambiguity could have suffered due to the informally described content model.

Finally, both the process engineer and the external reviewer rated an overall *content improvement of the development process model*, since it offered an approach to perform a requirements engineering approach for the particular domain of business information systems, which yet has been missing. However, they also stated that the significance of the improvement could only be evaluated by performing another case study, in which to apply the model at project level (see also the next case study).

6.2.3.4 Evaluation of Validity

In the following, we evaluate the validity of the case study with respect to the⁴

- *Construct Validity* that indicates the appropriateness of the study design with respect to the objectives and research questions.
- *Internal Validity* that indicates whether the study execution might have been influenced by side effects distorting the study results.
- *External Validity* that indicates to which degree the study design allows for generalisation of the results, i.e., that indicates the appropriateness of the case and subject selection with respect to the objectives and research questions.

Construct Validity. Regarding the construct validity, we cannot guarantee that the assessment criteria in the questionnaire completely represent our research questions. In addition, we could only simulate the process integration to some extent, since we did not integrate the approach during the case study into the processes of a real company. Thus, the effects of, e.g., organisational cultures could not be analysed. We see this as a minor threat, because we referred to a development process model that is practically used and aimed with the study at evaluating the technical applicability, which would barely change in an industrial setting.

Another threat is given by the subjective interpretation of the study participants during the assessment. However, we were particularly interested in their expert opinion and accordingly selected representative study subjects.

⁴ See also the contribution from Knapp [Kna09], which introduces the different types of validity in detail.

Internal Validity. The internal validity could be threatened by a bias towards the process integration approach, because the process engineer was involved in the development of the V-Modell XT and related approaches, which served, in part, to structure the process integration approach (see Sect. 5.2). This threat is, however, only minor, because the process engineer is neither benchmarking the V-Modell XT itself, nor his original approach.

As described in the validity procedures, we additionally performed a cross examination by referring to closed and open questions. Thus, each rating in the closed question includes a comprehensible rationale. Since we also involved an external reviewer for the assessment and the results of both interviews show same trends, we see the internal validity to be sufficiently ensured.

External Validity. As we exclusively evaluated the applicability of technical aspects of a process integration into a development process model (and omitted the last process integration step), we cannot generalise our findings to a process integration in an industrial setting. The applicability of the last step can only be assumed, because the process integration approach is also based on the experiences made during the development of our BISA reference model and its integration into a development process model of a particular company (Sect. 3.3.3). Still, even if we would have evaluated the process integration in another industrial setting, the results could not be generalised to general assertions, because organisational cultures are barely comparable.

6.3 Case Study 2: Customisation at Project Level

In this section, we present a case study that considers the application of our contributions at project level. In this case study, we take a practitioners perspective and evaluate the practical usage of our contributions: the artefact-based reference model (Chp. 4) to be applied in a project and the corresponding customisation procedure (Chp. 5.3).

In Sect. 6.3.1, we introduce the research objective and the industrial context of the case study. In Sect. 6.3.2, we introduce the study design, which we organise again according to the guideline proposed by Runeson et al. [RH09]. The results of the study are presented in Sect. 6.3.3, before discussing these results in Sect. 6.4.

The subsequently presented case study extends our contribution in [MFLPW11].

6.3.1 Research Objective and Context

Our objective is to evaluate our contributions at project level in an industrial context. Because our artefact-based reference model has been developed, evaluated, and used as the standard reference model at Capgemini TS (see the chosen validity procedure in Sect. 3.3.3), we gained practical experience there.

However, we have little empirical evidence on how the approach tackles particular problems beyond that, i.e., what the benefits and limitations of our approach are and whether our approach not only is yet another specific-purpose approach. Hence, we want to investigate whether our approach can be generally applied in real projects independent of the organisational culture of particular companies and whether it satisfies the objectives of the thesis stated in Sect. 1.2. We want to know whether our contributions satisfy the need of guiding the definition of a flexible RE process, which supports the creation of precise results.

6.3 Case Study 2: Customisation at Project Level

For this, we apply our artefact-based reference model and the customisation approach in a comparative case study. In this study, we analyse the actual RE process and the resulting artefacts in a development project at Siemens, which develops a traffic control system. We apply our artefact-based approach in the same context and compare both our approach and the previously used one with respect to possible improvements in the performed process and in the quality of the resulting artefacts.

Table 6.5 states our objective according to the goal definition template [WRH⁺02].

Table 6.5: Research objective and context.

Analyse	the integrated reference model and the customisation approach
for the purpose of	evaluation
with respect to their	quality improvements in comparison to the previously performed RE process
from the point of view of the	project participants
in the context of	a software development project

6.3.2 Case Study Design

In the following, we introduce the chosen case study design. After defining the research questions of the study, we describe how we select the case and the subjects. We then describe how we collect and analyse the data, before concluding with a discussion on the validity procedures.

6.3.2.1 Research Questions

The goal is to investigate the advantages and limitations in the application of our artefact-based BISA reference model and the customisation approach in an industrial project environment. In order to evaluate our objectives in offering a means to construct a flexible RE process that supports the creation of precise artefacts in response to individual project parameters, we formulate three research questions. These consider both the quality of the process and of the produced artefacts.

RQ 1. *Does the artefact-based approach improve the usability of the RE reference process?*

One aim of artefact orientation consists in achieving flexibility in the RE process, i.e., in supporting the application of the BISA reference model at project level in response to volatile project parameters. Thus, our first research question aims at analysing whether we can achieve an improvement in the usability of the overall RE reference process with respect to its application in a particular project.

RQ 2. *Does the artefact-based approach improve the syntactic quality of the created artefacts?*

Once we analysed the actual process for creating the artefacts with respect to individual project influences, we want to know whether the created artefacts are of higher syntactic quality due to the underlying artefact-based philosophy of our approach.

RQ 3. *Does the artefact-based approach improve the semantic quality of the created artefacts?*

Finally, for implementing the requirements it is not only important that the syntactic quality is high, but also that the requirements are stated correctly and sufficiently detailed, i.e., the semantic quality is high. Thus, although our contributions are specifically intended to improve the syntactic quality of the created artefacts, we still want to know whether the approach also improves their semantic quality.

6.3.2.2 Case and Subject Selection

In the following, we describe the case and the subject selection.

Case Selection. We apply the artefact-based reference model and the corresponding customisation approach to a software development project and repeat the RE process for a part of the system under consideration. Both the case and the subject selection are opportunistic, because we need a real development project and access to its participants and documentation.

Nevertheless, we choose a project that considers a similar application domain as the one in which our contributions have been developed but differs in its industrial context. Especially, we are interested in an evolving system, because Capgemini TS mostly replaces legacy applications completely by new systems.

To select a representative part of the system, we hold a discussion between the industry participants and researchers. We select a set of business domains (logically clustering potential processes and use cases, see Sect. 4.4.2) for which corresponding stakeholders are available. This way, the approach can be conducted entirely including the creation of both artefact types (the business and the requirements specification).

Subject Selection. We define three main groups of participants as study subjects:

1. *Industry Participants:* We assign experts from industry responsible for performing the (baseline) RE phase of the system development project under consideration. Ideally, they have different viewpoints on the requirements specification, e.g., product managers and developers.
2. *Researchers:* We additionally assign researchers, familiar with our approach, to take the role of the business analyst and the requirements engineer and to support the customisation procedure at the industry partner.
3. *External Reviewer:* We again call in an external reviewer not involved in the actual process in order to achieve an unbiased assessment of the produced specifications.

6.3.2.3 Data Collection Procedures

The collection of the data for the case study comprises the participation of the researchers in the RE workshops, as well as a concluding assessment of the performed process by the industry participants and the external reviewer. We first perform the process according to the customisation approach as part of a series of workshops between the researchers and the industry participants. Afterwards, we assess the process and the produced artefacts in direct comparison to the legacy process previously used in the company. The industry participants and the researchers do an *internal assessment* of the performed process and the produced

6.3 Case Study 2: Customisation at Project Level

artefacts. An external reviewer is called in especially for conducting an independent *external assessment* of the produced artefacts. Figure 6.7 depicts the procedure, which we explain in the following in detail.

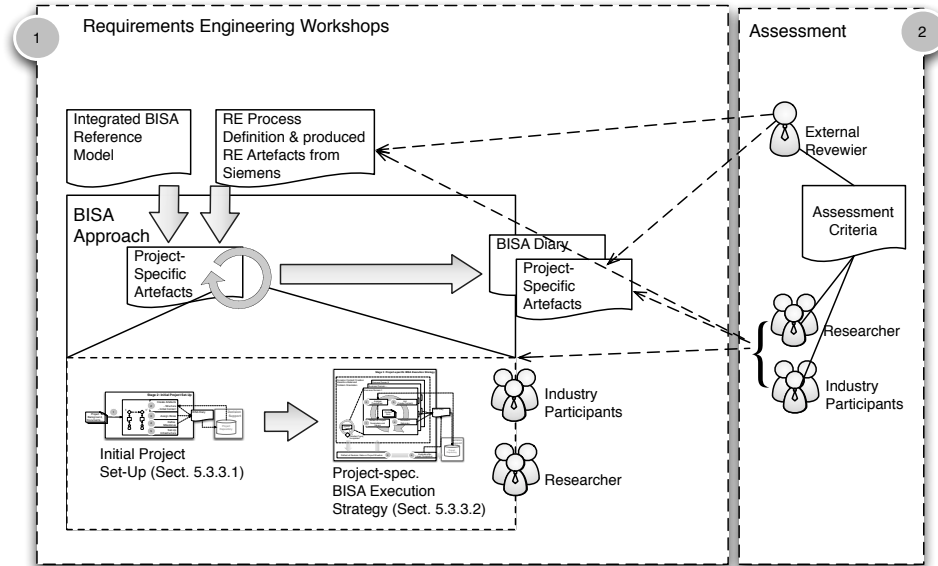


Figure 6.7: Overview of data collection procedure.

Requirements Engineering Workshops

We conduct the steps of the artefact-based customisation approach (Sect. 5.3.3) in a series of workshops. In these workshops, the researchers and the industry participants are present while the researchers are assigned to the BISA-specific roles.

Initial Project Set-Up. At a kick-off workshop, the researchers present the BISA reference model and the customisation approach. We customise the BISA reference model to initially set up the project. We select the artefacts to be created, decide on a document structure preferable by the industry participants, assign the roles, define the milestones for when to complete the artefacts, and finally agree on the used tool infrastructure. Restrictions on, e.g., the chosen milestones are initially documented as organisational requirements. Then, we discuss a set of specification documents, which we use as further input for the initial artefacts' content. This input includes a description of the business goals, the relevant stakeholders, and the business domains. We use the latter to organise the subsequent workshops.

Dynamic Content Creation. As part of the second customisation stage, we perform a workshop for each of the agreed business domains. In each of the workshops, we discuss the business processes of a domain and corresponding use cases. We use both the processes and the use cases to initially sketch the corresponding vision documents (business vision and the system vision). For the documentation of the requirements and the attributes, we use the template shown in appendix C. This template summarises the relevant attributes defined by the artefact model for the requirements. After each of the workshops, the researchers then further specify the artefacts' contents on the basis of the given information, i.e., they complete the

use cases or infer probable system quality requirements, which are presented and approved at the beginning of the next workshop. If making a particular decision relating to the content creation, they document this decision within the BISA diary.

Approval and Acceptance. One limitation of the case study known in advance results from the fact that the BISA reference model is not integrated into the chosen (industry) development process model⁵. Thus, we omit the last step in the customisation approach that considers the “reflection at a decision gate” on the consequences of potentially underspecified artefacts to further development activities that rely on those artefacts. We compensate this missing possibility of reflection by a formal approval and acceptance of the created artefacts. This means that after having created the content for each of the identified business domains, i.e., when reaching the agreed milestones, we jointly review the artefacts and the industry participants can reject or accept them with the possibility to demand further changes.

Assessment

We perform the assessment similarly as done in the first case study (see Sect. 6.2.2.3). As a preparation for the assessment, each industry participant and the external reviewer review both the previously documented specification documents and the BISA artefacts. As a guideline for this review, the reviewers get a questionnaire that will later be used in the interviews.

The goal of both interviews is to answer the questionnaire, which includes a set for different assessment criteria. Since the external reviewer has no insights into the actually performed process, the assessment criteria in his questionnaire is mostly reduced to the ones that consider the quality of the produced results. For each assessment criteria, we define a closed and an open question. In a closed question, we ask for the agreement to a given positivistic statement. It can be answered on a Likert-scale from *1 = I strongly disagree* to *8 = I strongly agree*. We choose again an 8-point scale to avoid that the interviewees check the middle. In the open question, we ask for their expert opinion and the interviewee answers as free text. This open question is also used for additional remarks and explanations regarding the selected grade on the Likert-scale. The condensed questionnaire with the statements of the closed questions is shown in Table 6.6.

To answer *RQ 1*, we ask for information on the execution of the process, respectively the customisation approach. Since only the industry participants and researchers were present while performing the process, only the industry participants will answer this part of the questionnaire. An exception is given by the assessment of the sustainability of the approach that is analysed by the external reviewer (only he can objectively analyse whether the process is reproducible on the basis of given documentation).

We answer *RQ 2* by assessing the syntactic structure of the specifications. This includes the structuring of the artefacts into topics, which, e.g., provide an easy understanding of traceability from high-level requirements to detailed ones. We additionally investigate the content in the artefacts with respect to cross-references between different content items, which serve, e.g., as background information or as a rationale.

Regarding *RQ 3*, we assess the actual content of the produced specifications. To rate the minimality and the completeness of the requirements, deep domain

⁵ We exclusively make use of the document templates generated with the V-Modell XT tool set.

6.3 Case Study 2: Customisation at Project Level

Table 6.6: Questionnaire for the assessment (condensed closed questions).

	Criteria	Statement
RQ 1	Ease of Use	The approach is clear and understandable.
	Sustainability	The process and the taken decisions are reproducible.
	Effectivity	All contents are used in subsequent development activities.
	Flexibility	The approach supports a flexible process in response to individual project characteristics.
	Productivity	The perceived productivity was high.
	Structuredness	The RE process is systematic.
RQ 2	Syn. Consistency	Elements in the specifications are used consistently.
	Complexity	The complexity in the cross-references is low.
	Syn. Completeness	All necessary syntactic elements are given in the reference model.
	Syn. Minimality	There are no unnecessary syntactic elements in the specifications.
	Modularity	The specification is organised in modules, separated according to certain topics.
	Traceability	Each requirement has a rationale.
	Ease of Perception	The specifications are well-suited to be understood by people not involved into the process.
RQ 3	Unambiguity	The requirements are stated unambiguously.
	Testability	The fulfilment of each requirement is measurable / testable.
	Sem. Completeness	All stakeholder needs are reflected by the specifications.
	Sem. Minimality	There are no needless requirements in the specifications.
	Sem. Consistency	There are no contradictory statements in the specification.

knowledge is necessary. Therefore, only the industry participants can answer these two questions.

6.3.2.4 Analysis Procedures

Same as in the first case study, statistical hypothesis testing is not applicable. Therefore, we present the results of the closed questions as a radar chart. We analyse the answers to the open questions qualitatively to further explain the answers of the closed questions, and to discuss the differences between the legacy process and the BISA-specific one.

6.3.2.5 Validity Procedures

To increase the reliability of the statements of the industry participants and, thus, the internal validity, we perform a group interview. Through the interaction between the group members, memories and experiences of the participants are stimulated. This way, they can produce insights that would be less accessible without this technique. Furthermore, the different group participants serve as quality control, because extreme opinions are filtered out by the participants [LT02].

Additionally, researcher triangulation is used to increase internal validity: in addition to the assessment of the specifications by the industrial participants (internal assessment), the assessment is done by a researcher not participating in the whole process (external assessment).

Moreover, methodological triangulation is used by asking both open and closed questions. Through the open questions the interviewees can express their opinion

more freely. On the other side, the closed questions force them to agree on one statement.

To mitigate the threat of a bias toward the BISA artefacts by the external reviewer, he is not involved in the study prior to the actual assessment. He is not allowed access to further documents, like background information, to ensure that his judgment relies only on the produced artefacts.

6.3.3 Case Study Results

After the description of the cases and subjects, we describe the assessment results and structure them according to the research questions. For reasons of confidentiality, we can not give detailed information on the (baseline) RE approach, the system under consideration, or illustrate the exact content of the involved artefacts.

6.3.3.1 Case Description

The case study is conducted with a department of Siemens AG. This department develops a traffic control system (TCS). The system is a hybrid of geographically distributed embedded controllers in traffic lights and a central information processing and monitoring system. To stick to the application domain for which our contributions have been developed, the analysed sub-system is such a monitoring system of the TCS.

The TCS is in production for more than 30 years, while each year a new release is developed. In each of those releases, Siemens conducts a development project with an RE process that takes three months. Table 6.7 briefly summarises selected background information on the envisioned development project in which we apply the BISA approach.

Table 6.7: Project background (condensed for reasons of confidentiality).

Project Scope	Re-design of TCS monitoring system to efficiently support the idealised business processes of the business domains <i>Operations, Maintenance, and Planning</i> of traffic signalling.
Customer	Market represented by communes.
Objectives	Reduction of customer complaints and service requests. Compliance to local juridical authorities. Compliance to international environments.

The development process model used by Siemens is based on the *Siemens Reference Process House*, which underlies the activity-based philosophy. The reference model is characterised by a set of milestones and corresponding methods, which are used to document the requirements in natural language into previously structured, self-contained *Excel Specifications*.

In the following, we describe the requirements engineering workshops performed in the introduced context.

Initial Project Set-Up. We agree on the project scope as summarised in Tab. 6.7 and assign the roles to the industry participants and the researchers (see also the following section describing the subjects). We define the milestones following time-boxing so that the produced artefacts can be, providing their acceptance,

6.3 Case Study 2: Customisation at Project Level

taken into account during the ongoing release development. We agree on documenting the business specification and the requirements specification. We structure both artefact types in a joint discussion according to the structuring possibilities illustrated in appendix B.1. The business demands analysis, and the risk status report are both omitted right from the beginning of the project due to, e.g., the chosen time-boxing and the weak access to the market needs (we document those reasons within the BISA diary). Furthermore, we sketch a project plan including appointments for the next requirements engineering workshops, which are organised according to the business domains depicted in the project scope of Tab. 6.7.

After the initial kick-off workshop, the researchers prepare the tool infrastructure according to the agreements made during the workshop. This includes the establishment of the CASE tools and of the project repository used to persist an initial set of project parameters to which we refer during the content creation of the artefacts. Regarding the used tool infrastructure, the BISA artefacts are recorded in *Microsoft Word* documents. To specify the content of the BISA artefacts, we extended the *Enterprise Architect, Version 7.5* with an add-in. This add-in is based on an UML profile that defines, in conformance to the concepts given in our artefact model, individual modelling shapes to specify, e.g., business processes and goal graphs. Appendix C gives further information on the used tool and the performed extension, as well as on the project repository and its creation.

Dynamic Content Creation. For the content creation of the specification documents, we consider the activities of three different business domains, which are supported by the monitoring system: *Planning*, *Operations*, and *Maintenance* (see Tab. 6.7). During operations, an operator is provided by the system with communication and controller states. In case of anomalies, these states are analysed and an initial fault clearance is initiated. Furthermore, the operators perform statistical analysis over a chosen period of time, and in case of traffic accidents they provide juridical evidence about the actual status of the corresponding traffic lights.

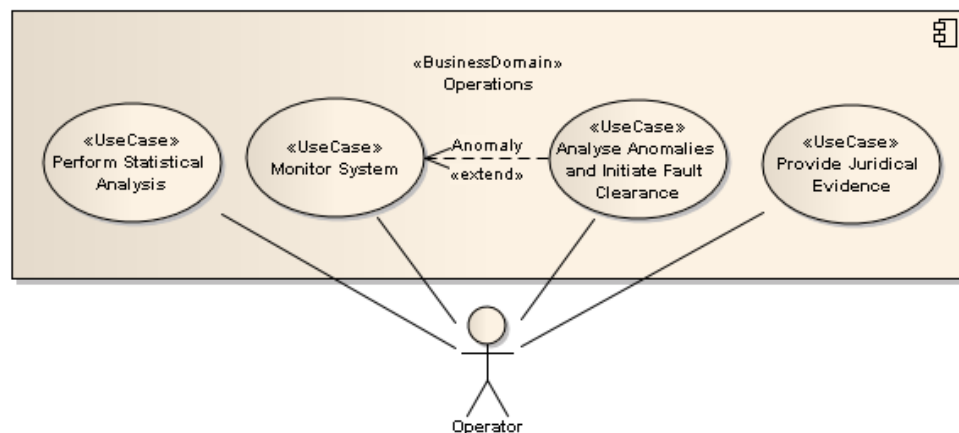


Figure 6.8: Exemplary (operations) use cases.

Figure 6.8 illustrates an exemplary excerpt of the resulting requirements specification, which contains the use case overview for the business domain “Operations”. Regarding decision taking during the content creation, we document each decision and the underlying project parameter in the BISA diary created with *Microsoft Word*. The BISA diary contains general content-related decisions as well

as (business) domain-specific decisions; for instance, we remove, in general, the time constraints from the requirements attributes to be captured, since the industry participants cannot commit themselves to concrete releases. An example for a domain-specific decision is, e.g., that we document business services instead of detailed business process models, because we decide for time-boxing.

In total, our project repository includes 31 project parameters (see appendix C). The BISA diary includes after the content creation as a whole 26 project parameters of which 8 are directly reused from the repository. This leads to a re-use ratio of approx. 30 % without a particular training phase for the repository involving several projects in a similar industrial context.

Approval and Acceptance. The researchers send the created artefacts (without the BISA diary) to the industry participants for an approval. After working in the given last feedback, the produced specifications are accepted and internally distributed within the development department.

6.3.3.2 Subject Description

As described in the subject selection, there are three groups of participants. In the group *industry participants* there are two roles, which we assign to three employees of Siemens AG.

The *Product Manager* is responsible for defining the requirements for the control and monitoring system from the customer/user viewpoint. He thereby represents potential customers. The *Project Lead* is responsible for broader management activities of the development department. Regarding requirements engineering, he is responsible for negotiating the requirements with the product manager.

The group of researchers consists of three software engineering researchers from the Technische Universität München with a special focus on requirements engineering. Apart from the author of the thesis, we choose Klaus Lochmann and Birgit Penzenstadler to participate in the RE workshops, since they also have detailed knowledge on RE in general and the BISA approach in particular.

Stefan Wagner of the Technische Universität München acts as external reviewer. He also has knowledge about RE. He is, however, exclusively involved in the assessment and excluded from the requirements engineering workshops.

6.3.3.3 Analysis Results

Figure 6.9 illustrates the results of the assessment as a radar plot, depicting the ratings of the closed questions. The upper radar plot summarises the ratings given by the industry participants, the lower radar plot summarises the ratings given by the external reviewer. We subsequently summarise the results taking into account the answers given in the open questions.

Usability of the Process (RQ 1). The BISA approach achieves an improvement of the usability in the process. The industry participants judged the BISA approach as *easier to use* due to its flexibility and its guidance given by the artefact model. In the interview, they explained that BISA defines a clear customised process. It needs, however, guidance during its customisation whereby deep knowledge in the approach (“skills”) is needed. The activity-based legacy process, used to produce the Excel specification, is more ad-hoc, since it offers no guidance with respect to individual project situations.

6.3 Case Study 2: Customisation at Project Level

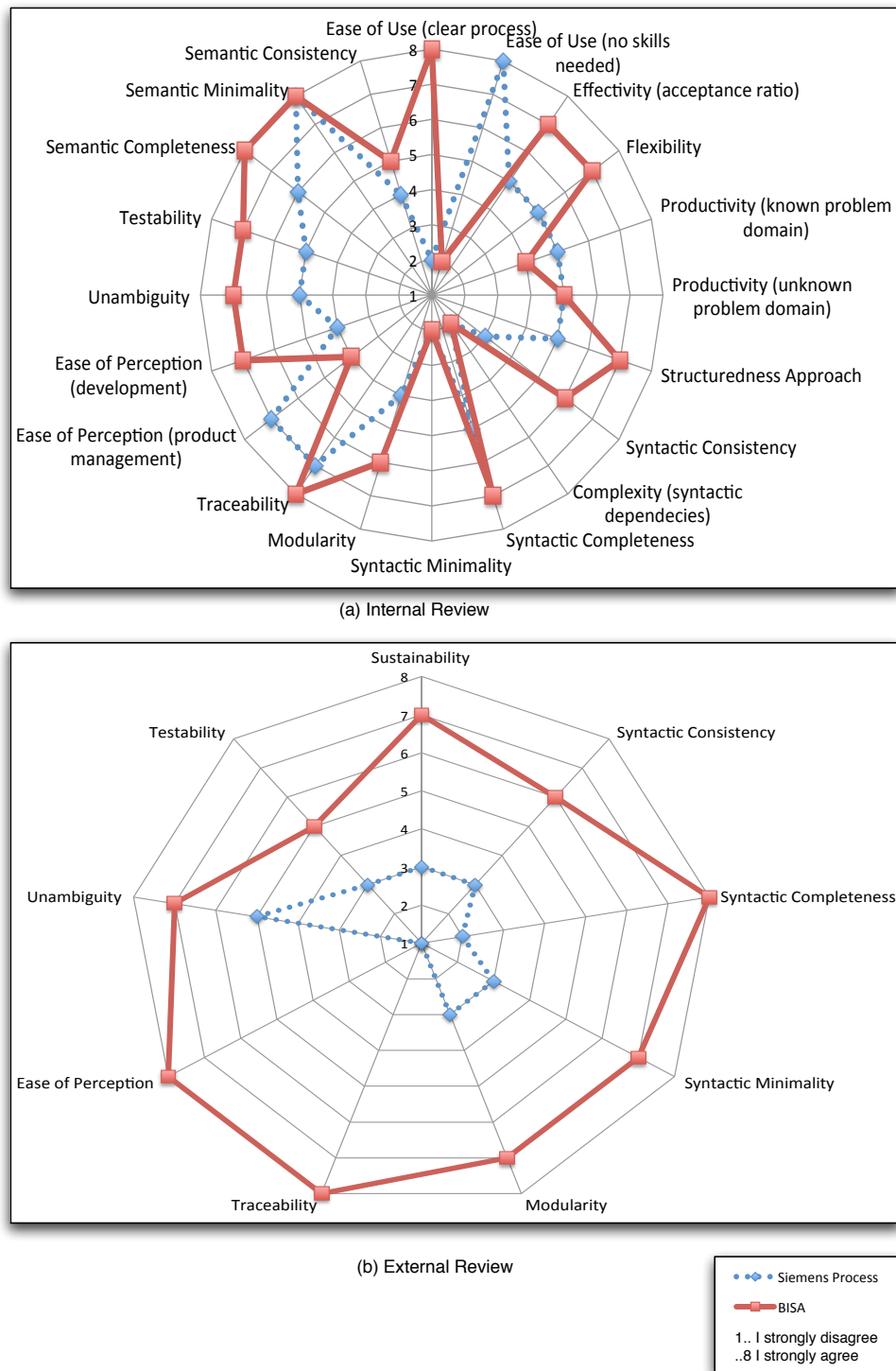


Figure 6.9: Analysis results shown as radar plots.

As our customisation approach gives guidance on structuring elicitation workshops according to logically related requirements clusters (business domains), they also saw an increase of the *structuredness* in the approach. The industry participants also stated an improvement in the *effectivity* as they noticed a higher accep-

tance ratio in the requirements, i.e., less requirements had to be re-adjusted and negotiated with corresponding architects after acceptance.

Regarding the *productivity*, they concluded that the BISA approach is more heavy-weight due to the artefact-based reference model, which demands for the specification of several topics in a certain syntactic quality. Still, they see this artefact-based approach to be beneficial if a previously unknown system is specified from scratch. However, if all participants already have a common understanding of the problem space (as in the department where the case study was conducted), then a more lightweight approach also seems adequate. Therefore, they judged productivity to be equal in both approaches.

Regarding the *sustainability* of the approach, the external reviewer stated that the BISA process could be reproduced. The reason is to be seen in the BISA diary, which documents all the made decisions leading to the created artefacts. In contrast to the BISA process, he could not reproduce the legacy process, because he was only confronted with the produced Excel specification.

Syntactic Quality of the Artefacts (RQ 2). Both the internal and external reviewers assessed the *syntactic quality* of the specifications. The internal reviewers judged the syntactic completeness to have slightly increased in BISA. They explained that in the Excel specification, they are able to define columns for all information needed. The restriction on Excel, however, excludes further possibilities for a syntactic representation, e.g., by referring to description techniques like the ones given in the UML. On the other hand, Excel offers filtering functions that can be used to aggregate needed information. BISA offers far more syntactic elements and proposes different description techniques for representing the contents, which are better suited for certain kinds of information.

This is also reflected by the judgment of the external review, who stated BISA to be substantially better regarding *syntactic completeness*. The reason is that the artefact model and the supported description techniques in BISA are more specific and therefore easier to understand to external people. The assessment is supported by the statement of the internal reviewers that the syntactic elements in Excel are inconsistently used, since they allow for an ambiguous interpretation. Thus, the meaning of the syntactic elements often remains unclear.

Both reviewers judged that there are less unused syntactic elements in BISA. The external reviewer noted, for example, that the column “state” in Excel is never used. One reason for the *syntactic minimality* of BISA is that the artefact model has been continuously customised.

The judgment on the *traceability* is different for both reviewers. The internal reviewers see a marginal increase in traceability in BISA, while the external reviewer assesses the traceability in the Excel specifications with the lowest grade and that of BISA with the highest. The reason for this difference can be found in the explanations the internal reviewers gave in the interview. They acknowledge that there is no rationale for requirements given in Excel. They know, however, that there are other documents in their company where the rationales are implicitly given. They further acknowledge that background information, like goals, are specified in BISA more comprehensively and structuredly. The difference in the judgment is further explained by the comments of the external reviewer. He could not discover any rationales for requirements in Excel, thus he judged the traceability with the lowest grade. In the BISA artefacts, however, he sees a clear top-down hierarchy given by the refinement notion in the artefact model, whereby implicitly necessary domain knowledge is made explicit.

6.3 Case Study 2: Customisation at Project Level

Semantic Quality of the Artefacts (RQ 3). Both the internal and external reviewers judged the *testability* of the requirements slightly better in BISA than in the Excel specifications. The testability, as well as the *unambiguity*, are improved in BISA, because the artefact model proposes a strict content model including several requirements attributes, such as acceptance criteria. Furthermore, both reviewers noticed that the free text formulations in the Excel specifications allows for more freedom in interpreting them. Again, the artefact model reduced this freedom and gives more guidance in the creation of precise artefacts. The internal reviewers, however, judged the resulting contents to be more perceptible by the development department rather than for the product management, since the content model demands for detailed specification of, e.g., architectural constraints.

The *semantic consistency* could be judged only by the internal reviewers, because of their domain knowledge. They rated the semantic consistency for the BISA artefacts slightly higher than for the Excel specification, because the dependencies in the artefact model are more suitable to find inconsistencies. The *semantic completeness* was stated to be higher in the BISA artefacts for a similar reason. The artefact model and the customisation approach supports a structured discussion of business needs and requirements, which were not considered before.

6.3.3.4 Evaluation of Validity

In the following, we evaluate the validity of the case study with respect to the construct validity, the internal validity, and the external validity⁶.

Construct Validity. Regarding the construct validity, we see the threat that the used questionnaire might not adequately represent the research questions. Although the questionnaire was developed jointly by all researchers participating in the study, it cannot be ensured that there are no topics missing.

Furthermore, we could not evaluate the customisation approach in full. In particular, we refer to the reflection at the decision gates, which we had to omit, because the approach was integrated into V-Modell XT (using in the case study the generated Microsoft Word templates), but not integrated into the development process model of Siemens. Hence, it was not possible to associate the BISA artefacts with further elements of the Siemens reference process, which would allow for a investigation of the effects of potentially underspecified BISA artefacts to other ones. However, we performed a formal approval and acceptance of the produced artefacts in which the industry participants reviewed the artefacts based on their knowledge about further going steps and artefacts in the development process.

Another possible threat to the construct validity is given by the used tool extension. The tools were extended according to the artefact model, but we used no conformance constraints to (automatically) ensure the syntactic conformance of the produced artefacts to the reference model. However, the case study was small enough to check this conformance manually. In addition, only the researchers who had detailed knowledge on the approach applied the artefact model.

A final threat to the construct validity is given by the subjective interpretation during the assessment. For instance, the industry participants rated the productivity equally in both approaches (with respect to unknown domains). This rating was given on the basis of their experiences and expectations on, e.g., a decrease of communication and negotiation needs and, in particular, an expected decrease of change requests. A statistically sound assertion would need, however, an observation of the project over its whole life cycle including also the investigation

⁶ See Sect. 6.2.3.4 introducing the three types of validity.

of change requests and their implementation. Hence, we cannot completely avoid this threat and have to rely on the expert judgement of the industry participants arising from the development and maintenance of the envisioned system. Still, we improve the participants' objectivity through the intensive discussions of the answers and their different view points.

Internal Validity. The internal validity could be threatened by a bias towards the BISA artefacts of the reviewers, because the external reviewer has detailed insights into the original development of the BISA approach. This threat is only minor for the internal reviewers, because they are comparing their legacy specification with the BISA artefacts.

Another threat to the internal validity could be that both specifications could have been created with a different effort. The better completeness and traceability of the BISA artefacts, for example, could be explained by more effort dedicated to it. However, this threat is seen as minor, because both approaches got the same rating on the productivity. As explained in the validity procedures, a researcher triangulation was done to mitigate reliability threats. When comparing the results of the internal and external assessment, we can see that the answers are differing, but have the same trend. This fact strengthens our confidence in the collected data.

Also, the methodological triangulation supports the internal validity. We could not find major contradictions between the explanations given in the open questions and the rating of the closed questions.

External Validity. Regarding the external validity, the major concern is the generalisability of the results, because we conducted this particular case study only in one company. From the viewpoint of the industry and research participants, however, a representative part of the system under consideration was selected.

6.4 Discussion and Conclusion

In the following, we give a summary of our conclusions and discuss the implications our case studies have. We then discuss the relation of our study to existing evidence, before concluding with a discussion of given limitations inherent in the chosen case study design.

6.4.1 Summary of Conclusions

The thesis aims at the establishment of an integrated, artefact-based reference model suitable to cover the needs of a customisable requirements engineering approach for the application domain of business information systems.

For this, we interpreted our meta model of Chp. 3 in an industrial context considering the analysis of business information systems to ensure the validity for the envisioned application domain (see Sect. 3.3.3). Based on this meta model and on its domain-specific interpretation, we established a customisation approach that considers the process integration of the reference model into a development process model and that customises the reference model at project level. Regarding the latter, we transferred the principles of decision support to the new artefact-based philosophy to guide in the creation of the artefacts in direct response to project characteristics that influence the degree of completeness in the artefacts.

6.4 Discussion and Conclusion

In the chapter at hand, we conducted two case studies as a means to evaluate our contributions in a qualitative manner. In the first case study (Sect. 6.2.2), we performed a process integration of our artefact-based reference model into the V-Modell XT following our procedure introduced in Sect. 5.2.2. In the second case study (Sect. 6.3), we customised our reference model to the needs of an individual industrial project environment following our procedure introduced in Sect. 5.3.3. We then compared the results to the RE approach previously used in the same industrial context.

Key Findings. With the case studies at hand, we showed that we achieved with our contributions the research objectives of the thesis. In particular, we showed with the first case study that our contributions are suitable to perform a structured process integration. The defined process integration approach seems to be, in general, usable while leading to a high quality process documentation. Although our approach showed to be generically defined to cover the specialities of different development process models, we could perform the analysis procedure and, finally, construct an integrated approach that achieves an overall content improvement of the envisioned development process model. The study participants rated the resulting process documentation overall positively, since we could, e.g., ensure the syntactic completeness of the development process model, while performing perceptible content improvements with respect to the needs of a particular application domain. However, although the BISA reference model seems to be unambiguously integrated into the development process model, the reviewers also stated that it still had to be evaluated at project level.

With the second case study, we performed such an evaluation in an industrial project environment. We performed a benchmark comparison in which we rated our approach and its customisation capabilities with respect to the previously used activity-based approach. The study participants rated our contributions to achieve an improvement of the syntactic quality in the results, but also in the semantic quality. We could achieve, for example, an improvement of the completeness and the consistency in the results, since our reference model guided in the creation of the artefacts w.r.t. the basic concepts and dependencies for the envisioned application domain. At the same time, the performed RE process proved to be flexible enough to cover the individual project characteristics.

We set up a project repository with 31 project parameters gained from our previously performed field study (see appendix C). We used those parameters to assist decision taking during the content creation of the artefacts and documented each decision in the BISA diary. We could achieve a re-use ratio in the parameters of approx. 30 %, while supporting with the diary a reproducible process that abstracts from complex specialities of the actually performed process.

Discovered Limitations. Despite the evidence given on the satisfaction of our research objectives, we also discovered limitations in our contributions.

The process integration approach needs further specialisation for chosen development process models into which the BISA reference model shall be integrated. This need considers, in particular, the analysis procedure and the critical interpretation of both meta models. The specialisation of our approach, e.g., by means of profiles, would also positively affect the unambiguity of the defined process integration steps.

At project level, we discovered two particular limitations in our contributions. One limitation is that our artefact-based reference model seeks more a technically affine audience. We do not, however, consider this as a severe limitation. For the

customer-specific stakeholders, we can increase the ease of perception of the created artefacts by choosing an appropriate syntax for their representation. For the development-specific roles, we see the necessary abilities to be a prerequisite for creating and controlling the artefacts (see Sect. 4.9). Another limitation is that our approach seeks more an application in unknown problem domains, e.g., reflected by the productivity.

6.4.2 Impact / Implications

In addition to considering more unknown problem domains, the establishment of artefact orientation is, in general, an elaborate task (see the development procedure described in Sect. 3.3.3). However, taking RE as a critical process that is mostly driven by uncertainty, it is important to support an integrated and flexible process design leading to high quality results. We could show that artefact orientation is promising to satisfy exactly those demands.

At project level, we showed, e.g., an increase of both the syntactic and semantic completeness and consistency of the artefacts, while those were flexibly and reproducibly created in response to individual project needs. We already argued in the fundamentals Sect. 2.6.4.1 that activity-based approaches and (situational) method engineering lacks of means to ensure the syntactic compatibility of (composed) methods potentially leading to inconsistencies in the artefacts, since the concepts incorporated within the different methods do not define explicit relationships to each other. In fact, we could show with the case study significant improvements of the syntactic consistency between the created artefacts and their contents.

From a practitioner's perspective, the results thus show that the effort necessary to establish an artefact-based RE approach should be justified by the resulting benefits. From a researcher's perspective, it seems promising to investigate more the field of artefact orientation to integrate existing methods for different application domains rather than exclusively focussing on the development of further isolated specific-purpose methods and description techniques with a limited area of action.

6.4.3 Relation to Existing Evidence

To the best of our knowledge, there does not exist any comprehensible case study that analyses specific activity-based approaches, artefacts-based ones, and, in particular, both in direct comparison. Hence, the case studies at hand close this gap in literature.

In fact, while most of the advantages of artefact orientation (stated in our context) still were based on arguments and, in parts, on assumptions, we could finally give evidence on those advantages.

6.4.4 Case Study Limitations

Inherent in case study research are given limitations, which can result in possible threats to the validity. First, both studies were single-project studies focussing each on one particular case and, thus, we had to cope with a small number of cases and subjects as a whole. The lack of data from multiple studies, which would allow empirically significant assertions on our observations, results in the threat that we cannot generalise from our findings in the cases studies [KPP94]. For example, although we showed in the second case study that the application of our artefact-based approach results in several improvements with respect to the previously used activity-based RE process, we cannot generalise this finding to the (to some

6.4 Discussion and Conclusion

extent) more philosophical question if artefact orientation in general “is always better” than activity orientation. Second, we limited the case studies not only in the number of cases and subjects, but also in the envisioned application domain.

However, our intention was not the empirically sound investigation of general benefits of artefact orientation, since we already have analysed and discussed these in Chp. 3 depending on concrete objectives that can be followed when deciding for artefact orientation. Instead, we wanted to give evidence on whether we satisfied our particular objectives with respect to the particular application domain for which our contributions have been developed.

In other words, we wanted to investigate whether we successfully tackled given shortcomings in available approaches and achieved the expected improvements in a particular setting, which we finally did.

CHAPTER 7

Summary and Outlook

Requirements engineering is a critical discipline that is driven by uncertainty. It is recognised that the basic knowledge about RE is often missing in practice [NSK00], whereby projects have little guidance in defining a systematic process that effectively copes with uncertain project situations. Available studies show that the RE process capability often suffers in practice [BHR03] implying a quality downstream of the produced artefacts, and finally potentially leading to project failures [Gro, KT07]. An improvement of RE capabilities is therefore known to be the key for successful and efficient projects [DC06].

While available studies give valuable insights into RE processes and their relations to project's success in general, we still had weak knowledge about the practical understanding of RE, potential problems in projects, and the consequences of those problems on the quality of the created artefacts. We thus performed an own *problem analysis* with a field study [MFWLB10, MFWL⁺12] (see Sect. 1.5). We analysed the establishment and the performance of RE processes in an industrial context considering different software development projects and their characteristics.

At organisational level, we showed that there is a missing awareness of an RE process that captures the basic concepts of the application domain. The discipline is treated in isolation neglecting the integration of its domain-specific concepts into concrete development process models. Hence, a reference model is missing that would enable an explicit transfer of domain knowledge among different projects. At project level, we discovered 31 project parameters that further barrier the appropriate choice of methods and their application during project execution and, thus, hamper the decisions on what artefacts to produce and on how to produce them in a syntactically consistent and complete manner. An observed consequence of missing company-wide references models and the diversity in the project influences is that project participants often act in a solution-oriented way. This is reflected in the quality of the produced artefacts being incomplete.

On the one hand, we thus could show that projects still have to face the problems existing since the late 1970's, where 46% to 50% of analysed IT development projects didn't pay sufficient attention to RE [Boe06, Hos87, Boe79]. On the other hand, we could track the problems down to particular underlying project characteristics and root causes, to be seen in the missing support of a flexible process that captures the concepts of an application domain in an integrated manner.

7.1 Summary of Contributions

Hence, motivated by our study, we defined in Sect. 1.2 the *research objectives* of this thesis. We aimed at defining an approach that

1. captures a model blueprint (reference model) of the basic modelling concepts of an application domain serving in development projects as orientation to create syntactically complete and consistent results.
2. gives guidance to:
 - a) integrate, at organisational level, the isolated reference model into the development life cycle by establishing associations between the elements of the reference model and the elements of a development process model.
 - b) customise the reference model at project level in direct response to individual, volatile project characteristics, which affect the completeness of the artefacts additionally supporting the reflection on the consequences of potentially underspecified artefacts on further development activities.

We argued that available related work in the area of activity orientation and method engineering could not sufficiently tackle the problems stated above. Available approaches rely on the idea of selecting and combining methods to a particular process rather than integrating and applying those methods considering their syntactic compatibility and the consistency in the resulting artefacts.

One promising possibility to achieve our objectives was to follow a process-neutral, artefact-based philosophy in which we concentrate on the artefacts and their dependencies rather than on the way of creating them. Within the artefact-based philosophy, the RE process would be flexibly established on basis of a set of artefacts that capture the basic concepts and relations of an application domain rather than on basis of a set of isolated methods and description techniques.

Using such a domain-specific reference model, we would be able to systematically integrate it into a development process model and to flexibly apply it at project level in direct response to individual project situations that affect the creation of the defined artefacts.

Motivated by this idea and the experiences we already made in artefact-based approaches in the area of development process models like the V-Modell XT, we aimed at tackling the problems stated above by defining an *artefact-based customisation approach for requirements engineering*, which yet is missing in literature.

In the following, we summarise our contributions and conclude with an outline of future work.

7.1 Summary of Contributions

In this thesis, we contribute an *artefact-based customisation approach for requirements engineering* considering the application domain of business information systems.

Inherent in the envisioned field of investigation is that our contributions cover a broader spectrum of related research areas and, thus, can be characterised as an extension and integration of relevant contributions on the basis of the artefact-based philosophy.

For this reason, it is important to ensure a scientifically valid research method. We covered this need by

1. analysing the field of artefact orientation and inferring a meta model with respect to our research objectives (Chp. 3),
2. interpreting this meta model for a particular application domain in an industrial context to ensure its validity (Chp. 4),

3. defining the customisation approach by transferring available related work to the artefact-based philosophy with respect to our meta model (Chp. 5), and by
4. evaluating our contributions in a qualitative manner with two case studies (Chp. 6).

We subsequently give a summary of our single contributions.

Analysis of Artefact Orientation and Inference of a Meta Model. In Chp. 3, we analysed the philosophy of artefact orientation with respect to our particular research objectives, since there is no common understanding and agreement on artefact orientation so far. We then developed a meta model for the philosophy unifying different notions of an artefact, which have before been treated in isolation. A detailed concept model gives guidance on the creation of artefacts by defining detailed concepts and relations of domain-specific description techniques. An abstract structure model supports the process integration and the customisation capabilities. We finally put those different artefact views into context of a development process meta model that now includes all further sub-models (e.g., milestones and roles) necessary to define an artefact-based process.

Interpretation of the Meta Model for the Domain of Business Information Systems. In Chp. 4, we interpreted our meta model for the application domain of business information systems. We developed the resulting artefact-based reference model for business information systems analysis (short: BISA) over the period of two years in an industrially supported research co-operation with the company Capgemini TS and continuously evaluated our contribution in this industrial setting (see Sect. 3.3.3 for further information). Thus, although our reference model is not the only possible interpretation of the meta model, we could ensure its validity for the application domain. As an additional validity procedure holds the definition of the artefact model itself. The model integrates, e.g., available, but isolated methods, and abstracts from established description techniques considered as relevant for the application domain during our previously performed field study.

Finally, the incorporated artefact abstraction model (Sect. 4.2), that defines domain-specific levels of abstraction and completion levels for the artefacts, allows for an objective determination of when artefacts are underspecified (incomplete) and the process therefore is solution-oriented what was barely possible before. We were able to effectively use this measure in an artefact-based customisation approach to assist (at project level) a conscious decision taking during the artefacts' creation in awareness of potentially underspecified artefacts.

Development of an Artefact-based Customisation Approach. Based on our understanding of artefact orientation and our domain-specific interpretation of the meta model, we presented an artefact-based customisation approach in Chp. 5. This customisation approach incorporates two major levels of abstraction in customisation. First, the organisational level considering the process integration of our reference model into a development process model. Second, the project level considering the customisation of our reference model according to the needs of particular project environments.

To establish the artefact-based customisation approach, we investigated related work and transferred available customisation approaches and principles to the artefact-based philosophy. A particular focus was put on the ideas given by situational method engineering and (content-centric) decision support systems.

7.1 Summary of Contributions

In contrast to this activity-based area of investigation, however, we characterised projects by the impact of project parameters on the creation of artefacts, instead of indirectly characterising projects by parameters impacting on the choice of methods for producing the artefacts. This characterisation supports awareness of what project characteristics to reflect on when creating the artefacts and, thus, we are able to appropriately assist decision taking. This decision taking was embedded into a concrete guidance on how to set-up a project (e.g., on how to initially create artefacts, assign the roles, and define the milestones) and on how to dynamically create the content of the artefacts in direct response to individual project parameters affecting the artefacts' completeness. For a reproducible decision taking, we additionally proposed a further artefact, the BISA diary, in which we document the decisions and transfer the project-specific experiences back to a central project repository (enabling knowledge transfer among different projects).

Due to the artefact model being integrated into a development process model, we were finally able to assist the objective reflection on potentially underspecified artefacts and their consequences on further development activities that rely on those artefacts.

Therefore, we are able to tackle the shortcomings given in available customisation approaches in supporting a flexible RE process that allows for the creation of precise and reproducible artefacts in direct response to individual project parameters.

Evaluation of our Contributions. Our contributions relied on an empirical problem analysis and a constructive research, which aimed at the development of the meta model, the artefact-based reference model, and the artefact-based customisation approach. Because of the development procedure chosen for the artefact-based reference model, we could also ensure the validity of our contributions for the application domain. However, while our contributions were motivated on the basis of arguments, we still had no empirical evidence on particular quality improvements for RE or, in general, whether we successfully achieved our research objectives. Hence, we performed two case studies that each evaluated our contributions in a qualitative manner (Chp. 6).

In the first case study, we evaluated the process integration of our approach by integrating it into the V-Modell XT. The involved study participants rated our process integration approach and the resulting modified development process model overall positively with respect to, for example, the structuredness, ease of use, and a low error-proneness, leading to an overall content improvement of the V-Modell XT.

In the second case study, we evaluated our approach at project level in an industrial context hosted by Siemens considering possible improvements with respect to the previously used activity-based RE approach. For this, we set up a project repository on the basis of project parameters, which we gathered from our previously performed field study. We then were able to show, for example, substantial improvements in the quality of the produced artefacts with respect to their syntactic and semantic completeness and consistency. In addition, the performed process showed to be flexible enough to cope with the individual project characteristics of that particular company.

Moreover, since there do not exist – to the best of our knowledge – comprehensible case studies that analyse activity-based approaches, artefact-based ones, or both in direct comparison, we could close with our case studies a gap in literature.

Finally, limitations of our contributions discovered in the case studies give us the opportunity to direct further research.

7.2 Future Work

We took first steps towards the integration and extension of different approaches on the basis of the artefact-based philosophy. Based on our results, we are able to understand the structure and principles of an artefact-based customisation approach, which is, on the one hand, necessary to support an integrated and flexible process, and, on the other hand, allows to guide in the creation of precise results.

However, our contributions are still first steps, whereby we subsequently discuss further possibilities of directing future research with respect to the artefact-based reference model, the artefact-based customisation approach, tool support, and further empirical research.

Artefact-based Reference Model. Our artefact-based reference model has been developed for the application domain of business information systems. It is, however, still unclear how our contributions relate to other domains, e.g., the one of embedded reactive systems.

As a first step, a comprehensive characterisation of different application domains would be necessary, since our domain characterisation arises (in the context of the thesis) from the architecture frameworks and development project characteristics considered during the development of our contributions. Based on a deeper understanding of the domains' principles independent of particular frameworks, we could identify differences and similarities of different domains and their effects on the content and structure of an artefact model. A next step should then consider the development of an artefact-based approach that is independent of a particular application domain, while including the domains characteristics into the customisation approach.

At the research group *Software & Systems Engineering* of the Technische Universität München, we are currently developing such a domain-independent artefact model that shall be customisable to different application domains. This approach aims at the integration of further methods, which are developed for different application domains. A first step into this direction is described in [BFI⁺09].

Artefact-based Customisation Approach. One limitation discovered in our case studies is that our process integration approach is described in a generic manner covering the specialities of different development process models into which our reference model is to be integrated. A lesson learned considers, in particular, the need of specialising the proposed analysis procedure by means of, e.g., profiles that give concrete guidance for the interpretation of and mapping between the envisioned meta models.

At project level, we further did not take into account a plan-based enactment, due to the limited scope on RE and because we focussed on the conceptualisation of decision support in the artefact-based context (see also the discussion in Sect. 5.1). However, the inclusion of mechanisms to generate a project plan and to effectively identify and manage deviations from the actual project execution to the planned one, would enable the usage of feedback loops. This, in turn, would support the deeper integration of project management activities with the analysis and development activities.

Furthermore, one limitation taken in advance, due to the strict focus on artefact orientation, was that we did not include the choice of different description techniques in the customisation approach, since we consider this as a restriction of its flexibility. In the context of the thesis, we allow the choice of any description technique

7.2 Future Work

for representing an artefact, as long as it covers the concepts and relations defined by the concept model. In the case studies, the choice of description techniques was performed intuitively. Our particular choice, however, negatively affected, in part, the ease of perception on product management level. Hence, the necessities and possibilities to further integrate the stakeholder-appropriate choice of description techniques into an artefact-based customisation approach needs further investigation.

Finally, we considered with our customisation approach, in general, the process integration and the project-specific application of the reference model, but disregarded a structural tailoring (see Sect. 2.6.1.1). During the development of our models (the meta model and the reference model), we did not consider means to support their further life cycle management, i.e., the inference and maintenance of variants. For example, we do not explicitly take into account the modification of our artefact model, e.g., by enriching selected content items with additional concepts, such as by enriching the business process model with additional dependency types. The support of such modifications is, however, important, whereby we need to further investigate the possibilities to support such modifications and the implications on the current meta model.

Tool Support. With the thesis at hand, we conceptualised artefact orientation for business information systems at three levels of abstraction: with the meta model, with the resulting artefact-based reference model, and at project level, where we create project-specific artefacts guided by our customisation approach.

Although our contributions and case studies sufficiently cover our research objectives by considering a pure conceptualisation, further possibilities for tool support should be investigated in the future. We consider, in particular, the tool-supported assurance of the conformance between our models at the different levels of abstraction.

First, the interpretation of our meta model for the application domain was performed manually. To further operationalise artefact orientation (also for other application domains), we are currently operationalising our meta model of Chp. 3 by means of a domain-specific language using the Process Development Environment (PDE)¹.

Second, when performing the second case study, we already used a prototypical tool extension to support the creation of the artefacts according to the artefact model (see appendix C). However, the artefacts created with our tool extension still had to be checked manually for conformance. Especially larger project environments with project participants not familiar with the artefact model would benefit from a CASE tool with respect to quality assurance, if the tool incorporates our artefact model in full including conformance constraints as proposed by Schätz [Sch01].

Empirical Research. Finally, our case studies give evidence on achieving our particular research objectives (see also the discussion in Sect. 6.4.4). We are, however, aware that we cannot generalise from our findings to perpetually valid advantages of artefact orientation towards activity orientation, since artefact orientation aims at achieving its specifically intended purpose, as it is the case with activity orientation.

Hence, although our case studies strengthen our confidence in the general advantages of artefact orientation when aiming at a flexible process design that supports

¹ Available at: <http://pde.codeplex.com/>

the creation of (syntactically) consistent and complete results, we need to extend our case study with further ones. Only this allows for empirically significant observations. By now, however, we contributed a first important empirical basis, which yet was missing in the area of artefact orientation, as well as in the area of activity orientation.

As a next step, we are currently operationalising our meta model for artefact orientation (see the foregoing paragraph). Based on this tool extension, we will be able to directly analyse the philosophy of artefact orientation reflected in our meta model in direct comparison with activity-based meta models, e.g., SPEM.

7.2 *Future Work*

Glossary of Terms used in the Thesis

Actor An actor is a unique role outside the system under consideration that is able to execute an action of a scenario and / or call an information system service. See also “Use Case”.

Application Domain An application domain considers the characterisation of an area (environment) in which systems of a particular family are used. The characteristics of an application domain influence the choice of (system) modelling concepts and the semantics of their notion. Note: The characterisation and classification of a particular application domain cannot be trivially expressed, because it also depends on the granularity of the view taken onto the used (family of) systems, usually reflected in architecture reference models, e.g., for business information systems in general or more specifically for service-oriented architectures.

Artefact An artefact is a deliverable that is produced, modified, or used by a sequence of tasks as part of one (development) phase and has value to a role. Artefacts are subject to version control and have a specific type. They are hierarchically structured into content items that define single areas of responsibility and that are output of a single task. Each content item encompasses at its lowest level of decomposition: (1) Concepts: a concept defines the elements and their dependencies of domain-specific description techniques used to represent the concern of a content item. Concepts have a specific type and can be decomposed to concept items, in which the differentiation is made if different items of a concept can be described with different techniques; (2) Syntax: the syntax defines a concrete language or representation that can be chosen for a specific concept; (3) Method: The method (respectively task) describes the sequence of steps that is performed in order to make use of a concept;

Balanced Problem Orientation A balanced problem orientation considers the appropriateness in the degree of problem orientation and means to act either in a problem-oriented or in a solution-oriented way, where possible and where necessary while having consciousness on the consequences of underspecified artefacts to other development activities (see also “(Project | BISA Execution) Strategy”); and “Problem Orientation”, respectively “Solution Orientation”).

Business Constraint A business constraint states a limitation placed on a solution, or an aspect of the current state that cannot be changed by the deployment of the new solution.

Business Driver A business driver represents internal and external factors, including individuals, knowledge and conditions, that initiate and support the design of business activities.

Business Domain A business domain groups a set of logically related business processes and their sub-processes. It defines a major area of the business that is in scope of the architecture, can be decomposed, and relates to one process owner.

Business Goal A business goal is a prescriptive statement of intent of one or more stakeholders. Business goals can be derived from other business goals, while we distinguish apart of business goals in general that concern financial and customer-related intents of the overall business in particular between (1) process-related goals and (2) IT-related goals.

Business Information System A business information system is an information processing system being structured in order to (at least partially) support business processes. It is characterised by (1) a boundary, respectively scope, (2) a (functional) behaviour that supports a set of business process and (3) further (also non-functional) properties. With the term “system”, we refer to all layers of a system that have the purpose of supporting selected business processes, including the systems’ architecture, used hardware, the underlying (network) infrastructures and one or more applications. Additional synonyms are “Information Systems (IS)” and “IT Systems”.

Business Information Systems Analysis (BISA) The phase BISA comprises two activities for an iterative and systematic creation of the two artefact types: (1) Business specification reflecting the needs of all relevant stakeholders towards the current and the future state of the business; (2) Requirements specification reflecting the needs and constraints towards information systems and their development, aligned with the business needs stated in the business specification.

Business Mission A business mission is a statement of direction and goal for the overall business.

Business Object A business object is a unique object of reality. It can be of material nature, such as a “Car”, or of immaterial nature, such as an “Order” that a waiter has in his mind.

Business Objective A business objective is a statement of intent concerning major achievements of an organisation typically within a specific time frame.

Business Process A business process is a cross-functional collection of all possible instances of related activities, performed across multiple business units, and contributing to at least one business goal. It is administrated by a process owner and serves to directly or indirectly produce a product or to provide a business service. The activities are triggered by an event (a change of state or a temporal event) and may use other activities to transform (produce, read, modify) one or more business objects and information objects. Business processes can be hierarchically decomposed until reaching the granularity of business tasks. Business tasks are partial units that are encompassed by a business process. Each business task consists of an ordered set of related atomic (not decomposable) process steps, while these are: (1) performed by exactly one user group that aims at achieving at least one process-related goal, and that (2) can be supported by an information system.

- Business Restriction** A business restriction is a not negotiable restriction towards the business processes, their structuring, and their interplay with information systems (see also business rule and business constraint).
- Business Rule** A business rule is an actionable directive that defines or constrains on behavioural aspects of the business and that supports a business goal. It is intended to assert business structure or to control or enable the activities of the business.
- Business Unit** A business unit is a recognised part of an organisation under management of at least one stakeholder.
- Business Vision** A business vision introduces into company-specific problems and defines the resulting scope of the business that shall underlie a change as part of a (development) project. The business vision steers the stakeholders into a common direction by summarising aimed objectives (needs) to be achieved, principles to be preserved, affected business capabilities to be gained and / or changed, and resulting (re-) structures of envisioned organisations.
- Conformance** A model *B* is conformant to another model *A*, if the concepts and relations used in *B* are a consistent and complete instance of *A*, including all imposed restrictions. Completeness requires that the concepts of the model considered conformant are present, while consistency requires that the relations must not be violated.
- Customisation (at Project Level)** Customisation considers the static and dynamic modification of a development process model during its application to a project, potentially in response to project parameters. Static customisation considers the initial creation of exemplars of a development process model ("enactment"). Dynamic customisation considers the subsequent content creation of the artefacts within the enacted process in direct response to project parameters that affect their creation and / or their degree of completeness.
- Development Process Model** A development process model is a standardised organisational reference model that abstracts from the idealised execution of a development project, including a description of artefacts (deliverables) to be produced, activities to be performed, and roles to be assigned.
- Enterprise** An enterprise is a set of organisations and their business units that share a set of common business goals and collaborate to provide specific products or business services to customers.
- Enactment** Enactment describes the procedure of applying a development process model to a particular project while the project entities and their relations being created (elements and relations found in the sub-models from which exemplars are created) are governed by the development process model. See also (static) "Customisation".
- External System** A external system is a system including one or more applications that directly or indirectly interacts with the envisioned information system in order to realise the work flow of a business process.
- Feature** A feature is a prominent or distinctive user-recognisable aspect, quality, or characteristic of an information system that is related to a specific set of requirements, whose realisation enable this feature. A feature groups a recognisable set of logically related requirements under one specific characteristic.
- Functional Requirement** A functional requirement specifies the demanded external system behaviour given by one stimulus, followed by one expected response, as part of a reaction to the stimulus, independent from the underlying realisation of this response.

Function A function is a capsule of (system) behaviour, defined by its external interface in terms of data and control flow as well as its internal implementation. Note that we use functions in addition to (atomic) services, respectively to a system action of a use case scenario and a functional requirement, as an explicit concept to represent the realisation of user-centric requirements as part of the (internal) specification of system functionality.

Goal See “Business Goal”.

Information Object An information object describes the information content of a business object that is processed and that can be reproduced by an information system (see also “Information System Object”).

Information System Object An information system object describes the system-side representation of an information object that can be processed in terms of being created, read, or updated.

Information System Requirements Information system requirements are requirements that exclusively describe demanded functionalities, properties and conditions of the system, its architecture and its environment.

Information System Service An information system service defines a piece of system’s functionality to support the realisation of process steps and / or business services. It can be hierarchically decomposed and maps an input (stimulus) to an output (response). Used synonyms are “Application Service” or “IT Service”.

Meta Model A meta model defines a set of (reference) models.

Method A method (or task) is a systematic construction procedure to combining description techniques for creating artefacts. A description technique covers a graphical convention (notation).

Migration Requirement Migration requirements are restrictions on the displacement or modification of one or more legacy systems, emphasising either aspects of (technical) migration steps or aspects of data migration.

Model A model is an abstraction of an object while not capturing every detail of the object in order to reduce the complexity in the model.

Obligation Obligations are agreements, propositions, and exclusions between the parties within a development project. They encompass demands towards a specific action that has to be taken by a party, independent of how the action is performed.

Organisation An organisation is an autonomous set of business units with defined boundaries.

Principle A (business) principle is a restricting directive, respectively statement of belief, approach or intent, which serves as a basis for directing the formulation of (business) architectures.

Problem Orientation Problem orientation considers the strategy of a continuous approval process in which the artefacts are analysed, refined, and quantified in consistency and compliance to the growing logical architecture (see also “Solution Orientation”).

Project A (development) project has a particular type and is characterised by a concrete set of artefacts being created by concrete team roles over a specific time frame (in-between milestones), motivated by a collection of project parameters. A project satisfies particular objectives of a customer and follows a particular project scope.

Project Execution A project execution considers the application (execution) of a development process (reference) model.

(Project | BISA Execution) Strategy A project execution strategy refers, in the context of the V-Modell XT customisation approach, to the planned order / way of reaching decision gates, i.e., points in times when particular products have to be finished. A project-specific BISA execution strategy, in turn, refers to the creation of particular BISA-specific artefacts in a particular degree of completeness in response to project parameters that influence the artefacts' creation. We distinguish a solution-oriented and a problem-oriented execution strategy, reflected in potentially underspecified artefacts as an outcome of particular decisions taken during the customisation procedure. See also "Customisation" and "Underspecified Artefact".

Project Requirement Project requirements are constraints towards the content and / or structure of selected artefact types and the process model (see also "Project").

Project Type A project type is a generalisation of projects with same or similar characteristics, i.e., a collection of chosen project parameters with same or similar values.

Process Integration Process integration considers the integration of approaches into a development process model by establishing the associations between the concepts of the approaches to be integrated and the concepts of the development process model.

Process Owner A process owner is an individual that has the responsibility for the performance of a business process in realising its business goals. He has the authority on the business process and its execution and the ability to make necessary definitions and changes.

Project Parameter A project parameter is an assessable condition from inside or outside the project that influences its execution, such as the availability of end users.

Quality Attribute A quality attribute reflects a composition of perceptible characteristics that are exhibited by a system and its environment due to its properties. An example is "Efficiency".

Quality Requirements See "System Quality Requirements".

Quality of Service The quality of service defines a contractual agreement on a set of service parameters including a set of service levels. A service parameter demands for a quality attribute for the service, like availability. A service level is a measurement that defines the expected value range for the chosen measurement, like 96%. Both the service parameter and the service level quantify the quality of service to a measurable or at least assessable level.

Reference Model A reference model is a model blueprint that defines (modelling) concepts and relations to be used as orientation for a particular application domain (see also "Application Domain").

Requirement A requirement is a demanded property of an information system or the process and environment related to the system's development and its integration, both to be accomplished by a development project.

Requirements Engineering Requirements engineering aims at describing a problem space as comprehensively as possible by comprising iterative and systematic approaches to define a requirements specification aligned to the needs of all relevant stakeholders.

Requirements Management Requirements management considers the procedure to administrate requirements artefacts over their life cycle.

Requirements Risk A requirements risk is a defected requirement that causes potential problems to the development activities and to the final software prod-

uct quality due to specific risk factors (root cause).

Role | Team Role A (team) role is a participant of a project with particular responsibilities and abilities.

Service Level Agreement See “Quality of Service”.

Scenario see “Use Case”.

Solution Orientation Solution orientation considers a strategy in which artefacts remain during the process underspecified. They are directly distorted by solution ideas as an outcome of design activities. The concepts that describe the problem space remain incomplete at the level of the business process hierarchy and / or the IT system hierarchy (see also “Problem Orientation”).

Stakeholder A stakeholder describes a unique individual, group, or organisation, that is related to an organisation and its business. It is actively involved in a project having its interests expressed by business goals that may be positively or negatively affected as a result of project execution or successful project completion.

Static Customisation see also “Enactment”.

System Overview A system overview specifies the system’s boundary by defining the interdependencies between the system and its future environment, given by the actors. These interdependencies define what actors intend to interact with the system under consideration and how they intend to do it.

System Quality Requirements System quality requirements constrain a system and its (development and usage) environment in order to support an activity, thus in order to satisfy a goals. System quality requirements directly constrain by the use of measurements specific behavioural (1) and structural (2) elements of the system and its environment in their properties and conditions in order to support the activities that rationale the requirement. System quality requirements affect at least one quality attribute.

System Vision The system vision defines the (sometimes contractual) basis for the following creation of the requirements specification. The vision summarises the results of the business specification that are in scope of the system, such as the business processes that have to be at least partially automatised, and aligns initial user-visible requirements to these results. It contains an outline of the envisioned information system’s capabilities and characteristics and aligns the defined stakeholders into a common direction by defining the application under consideration and its boundaries from a behavioural perspective.

Traceability Traceability considers the ability to describe and follow the life of a requirement, in both a (1) forward and (2) backward direction. The documented links between the requirements artefacts are defined as requirements traces.

(1): Forward tracing describes the investigation of traces forward to the artefacts that are accordingly produced.

(2): Backward tracing describes the investigation of traces backward to the artefacts’ rationale.

Underspecified Artefact A content item is underspecified if the created concepts and relations are in a state being not conformant to the corresponding concept types, i.e., incomplete. An artefact is underspecified if it has underspecified content items and these do not underlie further refinement, i.e., if the created concepts remain at the level of: (1) the business process hierarchy in case of the business specification and / or (2) the information system service hierarchy in case of the requirements specification.

Use Case A use case is a discrete, stand-alone and system-supported activity that is triggered by an event and performed by one or more actors in interaction with the system. It describes the observable behaviour and interaction of a system in response to an actor action. A single use case might encompass a collection of related structured scenarios, and one scenario is a specific instance of a use case. A use case is related to: (1) an actor that participates within a structured scenario; (2) a structured scenario as an ordered sequence of actions and interactions that occurs under certain conditions. These conditions describe the state of the system before the scenario is executed (precondition) and the state after the execution (postcondition); (3) an action as a unit that describes a specific behaviour within a scenario. It is either an actor action, or a system action. An actor action represents a process step that is realised by an actor, a system action a process step that is realised by the application; (4) an information system object as the system-side representation of an information object that is created or modified as a consequence of a system action.

User Group A user group is a representation of individuals with a specific proficiency directly performing a set of business tasks and potentially interacting with information systems.

APPENDIX B

Artefact Model

In this chapter, we illustrate the overall artefact model (the structure model and the content model) according to our previous contribution in [MFK09].

B.1 Structure Model and Possibilities for Arrangement

We illustrate in subsequent sections the possible structures of project-specific exemplars of the business specification and of the requirements specification.

B.1.1 Structuring of the Business Specification

Figure B.1 illustrates the structure of documents for recording the concepts of the business specification. The figure illustrates project-specific variations of the standard structure, which we introduced in Sect. 4.4.1.

We refer, in particular, to variations for

1. structuring the content according to project-specific business domains, as for example done in the business process model. In this case, we structure the single business processes for each of the corresponding business domains, which are defined by particularly named (additional) content items.
2. creating additional content items for single instances of selected concept types. For instance, we can describe all the business tasks of selected business processes within one general content item, or each of the business task separated by particular content items.

B.1 Structure Model and Possibilities for Arrangement

- 1 Introduction
 - 1.1 Overview
 - 1.2 Purpose
 - 1.3 References
 - 1.4 Scope
- 2 Business Vision
 - 2.1 Business Context
 - 2.1.1 Business Drivers, Objectives and Mission Statement
 - 2.1.2 Principles
 - 2.2 Project Scope
 - 2.2.1 Expert Problem Description
 - 2.2.2 Scope and Limitations
 - 2.3 Enterprise Overview
 - 2.4 Business Service Overview
- 3 Organisation Structure and Business Domains
 - 3.1 Organisation Structure
 - 3.2 Business Domains
- 4 Business Goals and Restrictions
 - 4.1 Business Restrictions
 - 4.1.1 Business Domain [Name] (Optional)
 - 4.2 Business Goals
- 5 Business Roles
 - 5.1 Stakeholder
 - 5.2 Process Owner
 - 5.3 User Groups
- 6 Business Capabilities
 - 6.1 Business Service Model
 - 6.1.1 Business Domain [Name]
 - 6.1.1.1 Business Service [Name]
 - ...
 - 6.1.1.n Business Service [Name]
 - ...
 - 6.1.n Business Domain [Name]
 - 6.2 Business Process Model
 - 6.2.1 Business Domain [Name]
 - 6.2.1.1 Business Process [Name]
 - 6.2.1.1.1 Business Task [Name] (Optional)
 - ...
 - 6.2.1.1.n Business Task [Name] (Optional)
 - ...
 - 6.2.1.n Business Process [Name]
 - ...
 - 6.2.n Business Domain [Name]
- 7 Business Information Model
 - 7.1 Business Objects
 - 7.1.1 Business Domain [Name] (Optional)
 - 7.1.1.1 Business Object [Name] (Optional)
 - ...
 - 7.1.1.n Business Object [Name] (Optional)
 - ...
 - 7.1.n Business Domain [Name] (Optional)
 - 7.2 Information Objects
 - 7.2.1 Business Domain [Name] (Optional)
 - 7.2.1.1 Information Object [Name] (Optional)
 - ...
 - 7.2.1.n Information Object [Name] (Optional)
 - ...
 - 7.2.n Business Domain [Name] (Optional)
- 8 Business Demands Analysis
- 9 Glossary

Figure B.1: Structuring the business specification.

B.1.2 Structuring of the Requirements Specification

Figure B.2 and B.3 illustrate the possible structure of the requirements specification according to similar variations as introduced in the previous section.

- 1 Introduction
 - 1.1 Overview
 - 1.2 Purpose
 - 1.3 References
 - 1.4 Scope
- 2 System Vision
 - 2.1 Summary of Business Specification
 - 2.1.1 Business Restrictions Overview
 - 2.1.2 Business Goals Overview
 - 2.1.3 Business Roles Overview
 - 2.1.4 Information Objects Overview
 - 2.1.5 Business Domains Overview
 - 2.1.6 Business Process Overview
 - 2.1.7 Business Service Overview
 - 2.2 Scope of Information System under Consideration
 - 2.2.1 System Overview
 - 2.2.2 External Systems
 - 2.2.3 Use Case Overview
 - 2.2.4 Information System Service Overview
 - 2.2.5 Features
- 3 Information System Requirements
 - 3.1 Actors
 - 3.2 Generic Scenarios
 - 3.3 Domain-specific Application Capabilities
 - 3.3.1 Business Domain [Name]
 - 3.3.1.1 Information System Service Model
 - 3.3.1.1.1 Information System Service [Name]
 - ...
 - 3.3.1.1.n Information System Service [Name]
 - 3.3.1.2 Use Case Model
 - 3.3.1.2.1 Use Case [Name]
 - 3.3.1.2.1.1 Functional Requirement [Name]
 - ...
 - 3.3.1.2.1.n Functional Requirement [Name]
 - ...
 - 3.3.1.2.n Use Case [Name]
 - Alternative
 - 3.3.1.2 Use Case Model
 - 3.3.1.2.1 Use Case [Name]
 - ...
 - 3.3.1.2.n Use Case [Name]
 - 3.3.1.3 Functional Requirements
 - 3.3.1.3.1 Functional Requirement [Name]
 - ...
 - 3.3.1.3.n Functional Requirement [Name]
 - ...
 - 3.1.n Business Domain [Name]
 - 3.4 Information System Objects
 - 3.4.1 Information System Object [Name] (Optional)
 - ...
 - 3.4.n Information System Object [Name] (Optional)
 - 3.5 System Quality Requirements
 - 3.5.1 System Quality Requirement [Name]
 - ...
 - 3.4.n System Quality Requirement [Name]
 - Alternative
 - 3.5.2 Business Domain [Name]
 - 3.5.2.1 System Quality Requirement [Name]
 - ...
 - 3.4.1.n System Quality Requirement [Name]
 - Alternative
 - 3.4.1 Usability Requirements
 - 3.4.1.1 System Quality Requirement [Name]
 - ...
 - 3.4.1.n System Quality Requirement [Name]
 - 3.4.2 Maintainability Requirements
 - 3.4.3 ...

Figure B.2: Structuring the requirements specification (1/2).

B.1 Structure Model and Possibilities for Arrangement

- Alternative
 - 3.4.1 Actor [Name]
 - 3.4.1.1 System Quality Requirement [Name]
 - ...
 - 3.4.n Actor [Name]
- Alternative
 - 3.4.1 Generic Scenario [Name]
 - 3.4.1.1 System Quality Requirement [Name]
 - ...
 - 3.4.1.n System Quality Requirement [Name]
 - ...
 - 3.4.n Generic Scenario [Name]
- 3.5 Architectural Constraints
 - 3.5.1 Logical Restrictions
 - 3.5.1.1 Architectural Constraint [Name]
 - ...
 - 3.5.1.n Architectural Constraint [Name]
 - 3.5.2 Technical Restrictions
 - 3.5.2.1 Architectural Constraint [Name]
 - ...
 - 3.5.2.n Architectural Constraint [Name]
- Alternative
 - 3.5.1 Generic Scenario [Name]
 - 3.5.1.1 Architectural Constraint [Name]
 - ...
 - 3.5.1.n Architectural Constraint [Name]
 - ...
 - 3.5.n Generic Scenario [Name]
- 4 Integrational Requirements
 - 4.1 Migration Requirements
 - 4.1.1 Migration Requirement [Name]
 - ...
 - 4.1.n Migration Requirement [Name]
 - Alternative
 - 4.1.1 Release [Name]
 - 4.1.1.1 Data Migration from Legacy Systems
 - 4.1.1.1.1 Migration Requirement [Name]
 - ...
 - 4.1.1.1.n Migration Requirement [Name]
 - 4.1.1.2 Displacement of Legacy Systems
 - 4.1.1.2.1 Migration Requirement [Name]
 - ...
 - 4.1.1.2.n Migration Requirement [Name]
 - ...
 - 4.1.n Release [Name]
 - 4.2 Deployment Requirements
 - 4.2.1 Deployment Requirement [Name]
 - ...
 - 4.2.n Deployment Requirement [Name]
- 5 Organisational Requirements
 - 5.1 Project Requirements
 - 5.1.1 Project Requirement [Name]
 - ...
 - 5.1.n Project Requirement [Name]
 - 5.2 Obligations
 - 5.2.1 Obligation [Name]
 - ...
 - 5.2.n Obligation [Name]
- 6 Requirements Risk Status Report
- 7 Glossary

Figure B.3: Structuring the requirements specification (2/2).

B.2 Comprehensive Content Model

The following three sections describe the overall concept model of the BISA reference model. We illustrated in Sect. B.2.1 the concept model of the business specification, in Sect. B.2.2 the one of the requirements specification, and in Sect. B.2.3 the dependencies between both.

B.2.1 Concepts of the Business Specification

Figure B.4 illustrates the overall concept model of the business specification that, in part, has been introduced in Sect. 4.4.2.

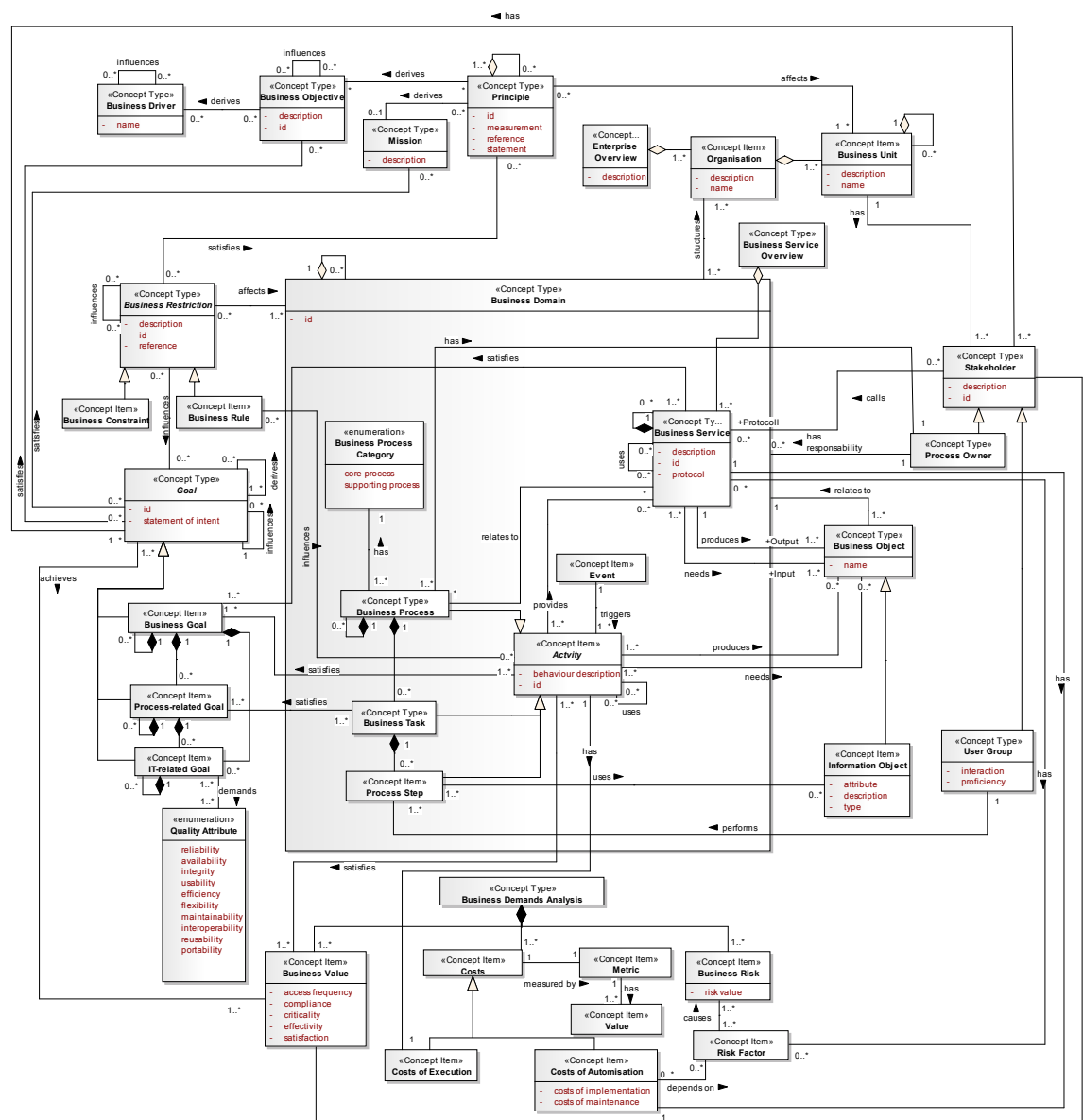


Figure B.4: Concept model of the business specification.

The diagram illustrates the ORE model, showing the relationships between various concepts and their attributes. The main components include:

- Concept Types:**
 - Feature:** Attributes: feature-id, description.
 - Use Case Model Overview:** Attributes: use case-id, description.
 - Information System Service Overview:** Attributes: name, description.
 - System under Consideration:** Attributes: name, description.
 - External System:** Attributes: name, description.
 - Scenario:** Attributes: id, description.
 - Use Case:** Attributes: use case-id, description.
 - Actor:** Attributes: id, primary actor (boolean).
 - Information System Service:** Attributes: id, description.
 - Collaboration Contract:** Attributes: protocol.
 - Quality of Service:** Attributes: name, description.
 - Service Parameter:** Attributes: name, description.
 - Service Level:** Attributes: value range.
 - Quality Attribute:** Attributes: reliability, availability, integrity, usability, efficiency, flexibility, maintainability, interoperability, reusability, portability.
 - Measurement:** Attributes: name, description.
 - Normative Reference:** Attributes: name, description.
 - Operative Background:** Attributes: name, description.
 - Value Range:** Attributes: name, description.
 - Variability:** Attributes: name, description.
 - Requirement Risk:** Attributes: id, description, risk score, risk impact, risk mitigation points, owner, status, time constraint, comments.
 - Risk Trend:** Attributes: name, description.
 - Risk Factor:** Attributes: name, description.
 - Deployment Requirement:** Attributes: name, description.
 - Obligation:** Attributes: name, description.
 - Project Requirement:** Attributes: name, description.
 - Migration Requirement:** Attributes: name, description.
 - Migration Aspect:** Attributes: legacy displacement, data migration.
- Concept Items:**
 - Structural Scenario:** Attributes: precondition, basic path (boolean).
 - Event:** Attributes: name, description.
 - Action:** Attributes: name, description.
 - System Action:** Attributes: name, description.
 - Actor Action:** Attributes: name, description.
 - Functional Requirement:** Attributes: name, description.
 - Information System Object:** Attributes: type, description, attribute.
- Relationships:**
 - derives:** Connects various concept types and items.
 - influences:** Connects various concept types and items.
 - participates:** Connects various concept types and items.
 - realizes:** Connects various concept types and items.
 - uses:** Connects various concept types and items.
 - calls:** Connects various concept types and items.
 - has:** Connects various concept types and items.
 - quantifies:** Connects various concept types and items.
 - measures:** Connects various concept types and items.
 - constrains:** Connects various concept types and items.
 - covers:** Connects various concept types and items.
 - causes:** Connects various concept types and items.
 - performs:** Connects various concept types and items.
 - exhibits:** Connects various concept types and items.

Figure B.5: Concept model of the requirements specification.

B.2.3 Dependencies between Concepts of the Business and the Requirements Specification

Figure B.6 illustrates the dependencies between both concept models illustrated in the foregoing sections.

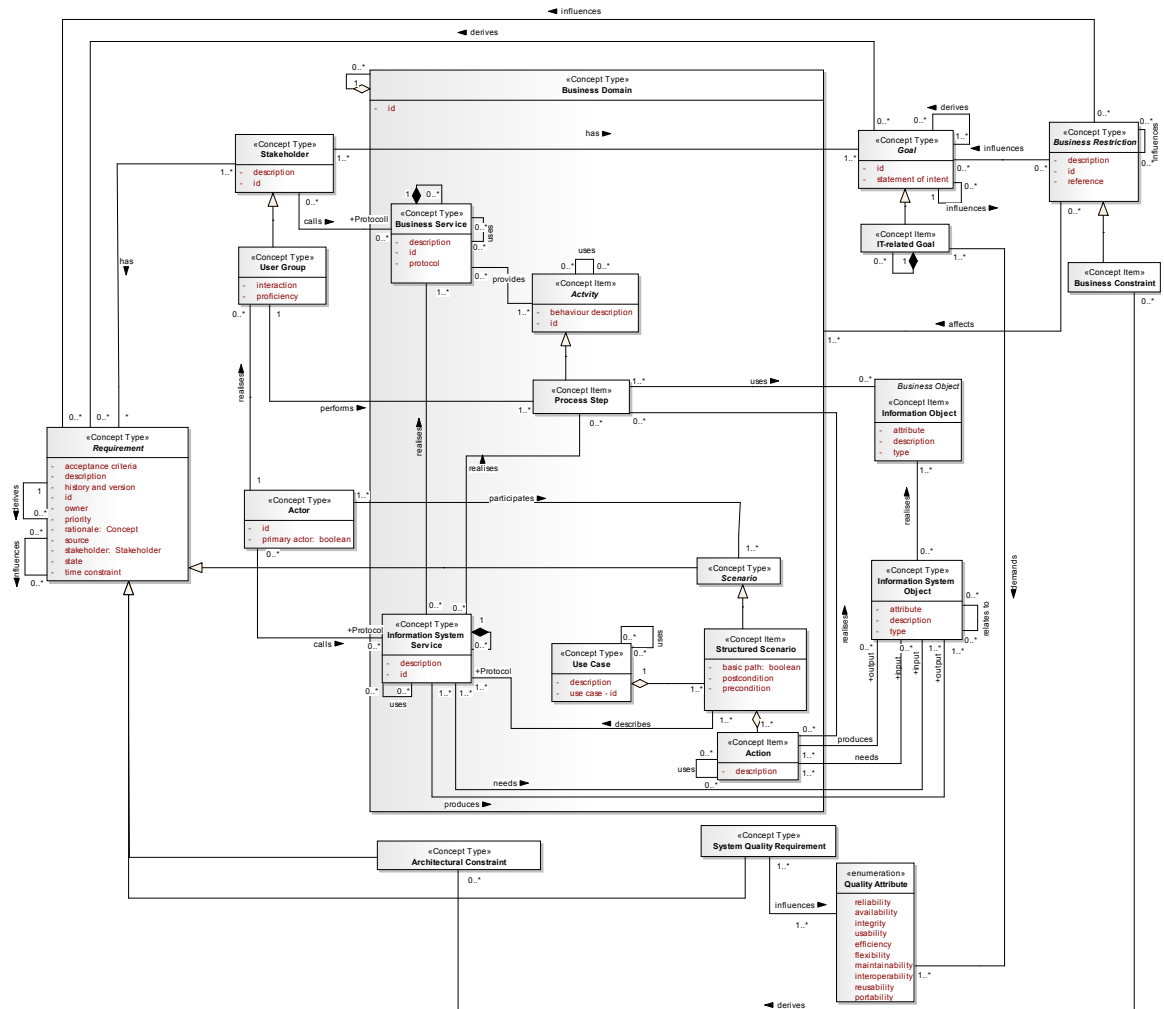


Figure B.6: Dependencies between concepts of the business and the requirements specification.

B.2.4 Dependencies between Concepts of the Requirements and a System Specification

The artefact model defined in the foregoing sections serves to specify behaviour and properties of systems taking a black-box view onto those systems. The dependencies to the concepts for modelling a logical component architecture depend, however, on the chosen architecture model, such as the one defined in UWE [KKZB08], or the one of the Technische Universität München [BFG⁺08].

For this reason, we give in this section only a brief overview of the dependencies of the concept model introduced in Chp. 4 and the concepts used to describe a

system architecture. With Fig. B.7, we take a generic view onto the dependencies between concepts used to describe non-functional aspects (the upper part of the figure), concepts used to describe functional aspects (the lower part), and concepts used to describe a system architecture (the centre of the figure).

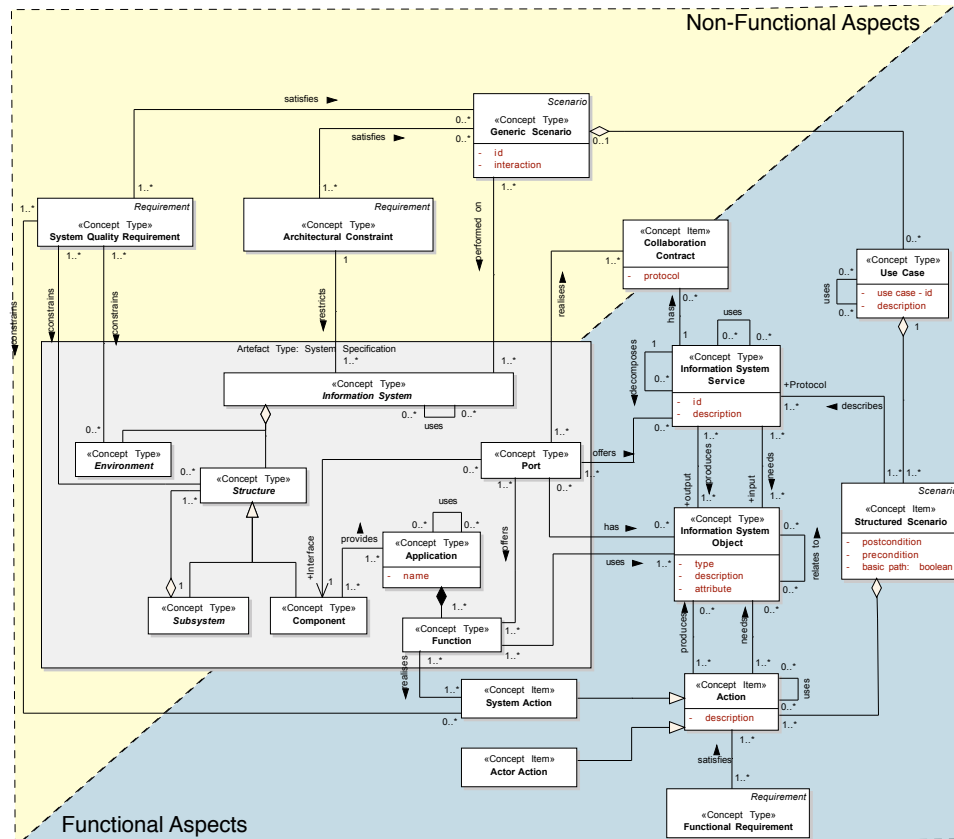


Figure B.7: Dependencies between requirements concepts and system concepts. Non-functional concepts are illustrated on upper part, functional concepts are illustrated on lower part.

With the system concepts, we refer to a generic understanding, as given in the V-Modell XT (see Sect. 2.5.3.2), in which systems provide functionality structured by a hierarchy of subsystems and components. A component offers functionality via (typed) ports [Sch08a] defining, in turn, possibilities to interact with the system, e.g., described as information system services. We make no distinction between different types of ports and interfaces, such as user interfaces.

To describe functionality, we use the concept *Function* as proposed by Schaetz: “A function is a capsule of system behaviour, defined by its external interface in terms of data and control flow as well as its internal implementation” [Sch08b]. A function is thus offered via ports, while it represents the *realisation* of at least one system action that is, in turn, described with structured scenarios. Similarly, the ports of single components describe what information system services are offered (respectively implemented) and thereby realise the protocol of an information system service (as part of a concrete instance of a service). As a consequence of this communication, the ports also describe what information system objects are processed, whereby the ports are typed according to these objects. How the dependencies between the services and the components exactly are, depends on the chosen granu-

larity of the components and the one of the services.

Regarding non-functional concepts, we make use of generic scenarios, which describe any kind of interaction with the system that is not functionally motivated by business processes. These interactions involve the system as a whole, including also its “environment” (e.g., the code). Referring to system quality requirements, they explicitly constrain one of the three main system concepts, i.e., structure, functionality, or environment. In contrast to system quality requirements, we associate architectural constraints with the structure of a system.

B.2 Comprehensive Content Model

APPENDIX C

Case Studies: Additional Background Information

In the following, we summarise additional information on the case study considering the evaluation at project level (Sect. 6.3).

Requirements Shell. We use during the elicitation workshops, where possible, the template illustrated in Fig. C.1.

Requirements Shell		
ID	What is the unique identifier of the requirement?	
(Original) Description	What is the requirement's description? (→ Give a brief informal description, which later on is specified in dependency to the assertion with respect to the concepts of the artefact model.)	
Owner	Who is responsible for resolving this requirement?	
Stakeholder	Who is the customer-side responsible individual or organisation that demands this requirement?	
Priority	What priority does the customer expect from the requirement?	
	Criticality	What impact would it have if the requirement is not realised?
	Satisfaction	How satisfied or unsatisfied would the stakeholder be if realising the requirement?
	Access Frequency	How often are related business processes performed?
	Effectivity	How well does the requirement support the stakeholder in achieving his business goal and / or business process?
	Compliance	Does the requirement result from a not negotiable constraint (such as a law)?
Source	Do there exist supplementary sources for the requirement, such as a document, an e-mail, or a protocol of a phone call or of a workshop?	
Rationale	What further requirements serve as a rationale for the requirement?	
History & Version	Is this requirement new? Was it changed? When was it changed? What changes have been made? What is the resulting new version?	
State	What is the actual state of the requirement?	
Acceptance Criteria	When is the requirement realised achieving the expectations of the stakeholder?	
Time Constraint (opt.)	Until when (or within what release) does the requirement have to be realised?	
Business Domain (opt.)	To what business domain is the requirement related?	

Figure C.1: Template used during elicitation workshops.

The content of the template captures the basic concepts of a requirement (see appendix B.2.2). During the elicitation workshops, we informally document the requirements with a particular focus on the context information (attributes) and further given background information, before specifying the content (“description”) in full with respect to the concept model.

Tools used in the Case Study. We document the BISA artefacts as Microsoft Word documents, which we structure according to the artefact model. To define the content of the artefacts, we refer either to natural language following the concepts defined in the artefact model or to models created with a modelling tool. For the latter, we use the *Enterprise Architect, Version 7.5 (Sparx Systems)* and modify an extension, which we initially developed in the research co-operation with Capgemini TS (see Sect. 3.3.3). We subsequently give a brief explanation of the extension used in the case study. The enterprise architect allows, in general, for the definition of

1. templates used to define a standard package structure to organise the produced models.
2. (UML) profiles used to define the elements and relations necessary to create those models including graphical conventions.

Figure C.2 illustrates an exemplary excerpt of the extension made for the artefact type *Business Specification*.

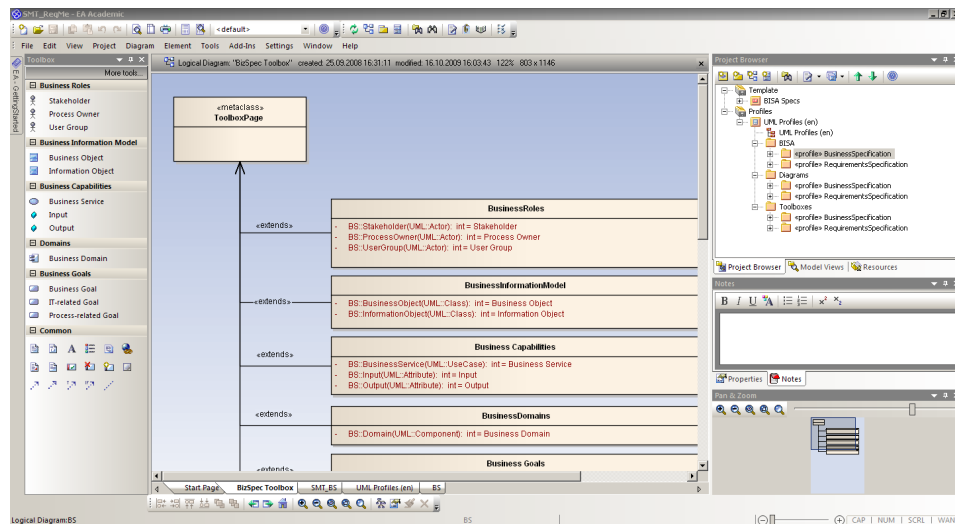


Figure C.2: Screenshot: Enterprise Architect extensions.

As we can see in the project browser on the right side of the illustration, the definition of a profile includes the actual *UML Profile* itself (here named “BISA”), a set of *Diagrams*, and a set of *Toolboxes*.

The profile defines the representable elements and relations, which we define according to our concept model. The diagrams serve as containers to create the models, e.g., a business information model. Toolboxes define the graphical conventions used to create the single diagrams; for instance, the elements used to create the business information model.

In the centre of the figure, we illustrate the exemplary toolbox definition for the elements of the business specification. At the example of the business information

model, we make use of graphical elements (“shapes”) available for the specification of UML class diagrams to specify, e.g., business objects. These elements are then illustrated during their creation as classes with the corresponding stereotype. Similar extensions are made for the other concepts defined by BISA. For example, we make use of the graphical conventions used for the elements in UML state charts to represent the elements in goal graphs. The resulting tool boxes are shown on the left side of the figure.

After defining the profile and the templates for the business specification, as well as for the requirements specification, both are exported as XMI and made available for the creation of new projects in the Enterprise Architect by the use of an in-built importer (“Model Driven Generation-Wizard”).

For the case study, we then create after the first kick-off workshop a new project and make use of the defined templates and the profile. The BISA toolboxes then are available for creating for each BISA content item selected diagrams by the use of the pre-defined graphical conventions, and each diagram is persisted in the templates according to the BISA structure model.

Project Repository. During the case study, we set up a project repository in order to guide in the content creation of the artefacts following the customisation approach of Sect. 5.3.3. The project repository is informally defined in a spreadsheet and made available to the researchers who directly create the artefacts. For reasons of complexity, we do not define a detailed project characterisation by means of, e.g., project types and only define a set of project parameters and their effects on the creation of the artefacts. Furthermore, we choose naming conventions for the project parameters that already express concrete values.

To set up the repository, we refer to the results of our previously performed field study [MFWL⁺12] (see Sect. 1.5) in which we collected a set project parameters and their influence on the creation of selected artefacts. Since we used during this analysis an artefact model that differs from the one contributed in the thesis at hand, we transfer the study results to our artefact model.

Table C.1 on page 234 illustrates the resulting project parameters and their influences on our artefact model. The columns in the table illustrate selected content items of the artefact model. The rows define the project parameters, which we organise by three parameter categories. The category *Customers’ Domain* groups parameters that arise from the industrial sector or from circumstances of a customers’ organisation, such as stakeholder-specific characteristics. The category *System under Consideration* groups parameters that arise from the envisioned family of systems and corresponding characteristics. The third category *Cross-Cutting Process Aspects* groups further parameters that can be allocated to the project-specific process. The resulting cells of the table define the actual impact scales. We use “+” to illustrate that the parameters argue for the creation of the content item and “−” to illustrate that the parameter hamper the creation of the content items.

For example, a high team distribution argues for the creation of the business and system vision (for scoping), as well as for the creation of glossaries. Another example is given by weak technical abilities of the stakeholders, which negatively affect the creation of detailed system quality requirements. At the same time, this parameter argues for the definition of a requirements risk status report. Further information on the identified project parameters and their general effects can be taken from [MFWL⁺12].

Table C.1: Simplified project repository used in case study.

[illegible]

Bibliography

- [Ale03] I. Alexander. Misuse Cases: Use Cases with Hostile Intent. *IEEE Software*, 20(1):58–66, 2003.
- [AW05a] A. Aurum and C. Wohlin. Aligning Requirements with Business Objectives: a Framework for Requirements Engineering Decisions. In *Proceedings Requirements Engineering Decision Support Workshop (RE-DECS'05) held in Conjunction with the 13th IEEE International Conference on Requirements Engineering*, page N/A, 2005.
- [AW05b] A. Aurum and C. Wohlin, editors. *Engineering and Managing Software Requirements*. Number ISBN-13: 978-3642064074. Springer-Verlag Berlin, 2005.
- [BA98] B. Boehm and E. Alexander. Software Requirements Negotiation: Some Lessons Learned. In *Proceedings of the 20th International Conference on Software Engineering (ICSE 98)*, pages 503–506, Washington, DC, USA, 1998. IEEE Computer Society.
- [Bal98] H. Balzert. *Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*. Number ISBN-13: 978-3827411617. Spektrum, Akademischer Verlag, 1998.
- [BBK⁺78] B. Boehm, J.R. Brown, H. Kaspar, M. Lipow, G.J. Macleod, and M.J. Merrit. *Characteristics of Software Quality*. Number ISBN-13: 978-0444851055. Elsevier Science Ltd North-Holland Pub. Co., 1978.
- [BCK03] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. Number ISBN-13: 978-0321154958. Addison-Wesley Longman Publishing Co., Inc., 2003.
- [BCVP06] S.j. Bleistein, K. Cox, J. Verner, and K.T. Phalp. B-SCP: A Requirements Analysis Framework for Validating Strategic Alignment of Organizational IT based on Strategy, Context, and Process. *Information and Software Technology*, 48(9):846–868, 2006.
- [BEJ06] R. Buschermöhle, H. Eekhoff, and B. Josko. *Success – Erfolgs- und Misserfolgsk Faktoren bei der Durchführung von Hard- und Softwareentwicklungsprojekten in Deutschland*. Number ISBN-13 978-3-8142-2035-2. BIS-Verlag der Carl von Ossietzky Universität Oldenburg, 2006.
- [Ber08] A.J. Berre. Service oriented architecture Modeling Language (SoaML) - Specification for the UML Profile and Metamodel for Services (UPMS). Technical report, Object Management Group (OMG), August 2008.

Bibliography

- [BFG⁺08] M. Broy, M. Feilkas, J. Grünbauer, A. Gruler, A. Harhurin, J. Hartmann, B. Penzenstadler, B. Schätz, and D. Wild. Umfassendes Architekturmodell für das Engineering eingebetteter Softwareintensiver Systeme. Technical Report TUM-I0816, Technische Universität München, 2008.
- [BFH⁺10] M. Broy, M. Feilkas, M. Herrmannsdoerfer, S. Merenda, and D. Ratiu. Seamless Model-Based Development: From Isolated Tools to Integrated Model Engineering Environments. *Proceedings of the IEEE*, 98(4):526–545, 2010.
- [BFI⁺09] M. Broy, A. Fleischmann, S. Islam, L. Kof, K. Lochmann, C. Leuxner, B. Penzenstadler, D. Mendez Fernandez, W. Sitou, and S. Winter. Towards an Integrated Approach to Requirements Engineering. Technischer Bericht TUM-I0935, Technische Universität München, 2009.
- [BGK⁺07] M. Broy, E. Geisberger, J. Kazmeier, A. Rudorfer, and K. Beetz. Ein Requirements Engineering Referenzmodell. *Informatik Spektrum*, 30(3):127–142, 2007.
- [BGN06] M. Brenner, M. Garschhammer, and F. Nickl. Requirements Engineering und IT Service Management - Ansatzpunkte einer integrierten Sichtweise. In H.C. Mayr and R. Breu, editors, *Modellierung 2006*, volume P-82 of *Lecture Notes in Informatics (LNI)*, pages 51–66. Gesellschaft für Informatik (GI), 2006.
- [BHR03] S. Beecham, T. Hall, and A. Rainer. Software Process Improvement Problems in Twelve Software Companies: An Empirical Analysis. *Empirical Software Engineering*, 8(1):7–42, 2003.
- [BHR05] S. Beecham, T. Hall, and A. Rainer. Defining a Requirements Process Improvement Model. *Software Quality Journal*, 13(3):247–279, 2005.
- [BKM07] M. Broy, I. Krüger, and M. Meisinger. A Formal Model of Services. *ACM Transactions on Software Engineering Methodology (TOSEM)*, 16(1):1–40, 2007.
- [BLMF⁺10] M. Broy, C. Leuxner, D. Mendez Fernandez, L. Heinemann, B. Spanfelner, R. Schlör, and W. Mai. Towards a Formal Engineering Approach for SOA. Technischer Bericht TUM-I1024, Technische Universität München, 2010.
- [Boe79] B. Boehm. Software Engineering: As it is. In *Proceedings of the 4th international conference on software engineering (ICSE 79)*, pages 11–21, Piscataway, NJ, USA, 1979. IEEE Press.
- [Boe03] B. Boehm. Value-Based Software Engineering. *ACM SIGSOFT Software Engineering Notes*, 28(2):3–15, 2003.
- [Boe06] B. Boehm. A View of 20th and 21st Century Software Engineering. In *Proceedings of the 28th International Conference on Software engineering (ICSE 06)*, pages 12–29. ACM Press, 2006.
- [Boe08] J. Boegh. A New Standard for Quality Requirements. *IEEE Software*, 25(2):57–63, 2008.
- [BPKR09] B. Berenbach, D. Paulish, J. Kazmeier, and A. Rudorfer. *Software & Systems Requirements Engineering: In Practice*. Number ISBN-13: 978-0071605472. McGraw-Hill Osborne Media, 2009.
- [BR87] V. R. Basili and H. D. Rombach. Tailoring the Software Process to Project Goals and Environments. In *Proceedings of the 9th International Conference on Software Engineering (ICSE '87)*, pages 345–357. IEEE Computer Society Press, 1987.

- [Bri96] S. Brinkkemper. Method Engineering: Engineering of Information Systems Development Methods and Tools. *Information and Software Technology*, 38(4):275–280, 1996.
- [Bro03] M. Broy. Service-Oriented Systems Engineering: Modeling Services and Layered Architectures. In H. König, M. Heiner, and A. Wolisz, editors, *Formal Techniques for Networked and Distributed Systems-FORTE*, pages 48–61. IFIP International Federation for Information Processing, Springer-Verlag, 2003.
- [Bro05] M. Broy. Service-Oriented Systems Engineering: Specification and Design of Services and Layered Architectures. In M. Broy, J. Gruenbauer, and T. Hoare, editors, *Proceedings of the NATO Advanced Study Institute on Engineering Theories of Software Intensive Systems*, volume 195, pages 47–81. Springer-Verlag, 2005.
- [Bro06] M. Broy. Requirements Engineering as a Key to Holistic Software Quality. In A. Levi, E. Savas, H. Yenigun, S. Balcisoy, and Y. Saygin, editors, *Proceedings of the 21th International Symposium on Computer and Information Sciences (ISCIS 2006)*, volume 4263, pages 24–34. Springer-Verlag Berlin, 2006.
- [BS01] M. Broy and K. Stølen. *Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement*. Number ISBN-13: 78-0387950730. Springer-Verlag Berlin, 2001.
- [Bus03] Business Modeling & Integration Domain Task Force. Business Process Model and Notation (BPMN) V2.0. Technical Standard formal/2011-01-03, Object Management Group (OMG), 2003.
- [BW07] B. Berenbach and T. Wolf. A Unified Requirements Model; Integrating Features, Use Cases, Requirements, Requirements Analysis and Hazard Analysis. In *Proceedings of the 2nd International Conference on Global Software Engineering, 2007 (ICGSE 2007)*, pages 197–203. IEEE Computer Society, 2007.
- [BWHW05] C. Braun, F. Wortmann, M. Hafner, and R. Winter. Method Construction - A Core Approach to Organizational Engineering. In *Proceedings of the 20th ACM symposium on Applied computing (SAC '05)*, pages 1295–1299. ACM New York, NY, USA, 2005.
- [BWSV08] S. Brinkkemper, S. Weerd, M. Saeki, and J. Versendaal. Process Improvement in Requirements Management: A Method Engineering Approach. In B. Paech and C. Rolland, editors, *Proceedings of the 14th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 08)*, volume 5025/2008, pages 6–22. Springer-Verlag Berlin, 2008.
- [CCR⁺95] G. Campbell, M. Cochran, S. Rose, L.P. Gates, and K.A. Johnson. A Tailorable Process for Systems Engineering (V.01.00.05). Technical Report A868092, Virginia Center of Excellence for Software Reuse and Technology Transfer, 1995.
- [CHS08] A. Classen, P. Heymans, and P.-Y. Schobbens. What’s in a Feature : A Requirements Engineering Perspective. In J. Fiadeiro and P. Inverardi, editors, *Proceeding of the 11th International Conference on Fundamental Approaches to Software Engineering (FASE 08) in Conjunction with ETAPS 08*, number 4961 in FASE/ETAPS, pages 16–30. Springer-Verlag Berlin, 2008.
- [Cle05] D.I. Cleland, editor. *Project Management Field Guide*. Number ISBN-13: 978-0471292067. John Wiley and Sons, Inc., 2005.

Bibliography

- [CM78] J.P. Cavano and J.A. McCall. A Framework for the Measurement of Software Quality. *SIGSOFT Software Engineering Notes*, 3(5):133–139, 1978.
- [CNV09] K. Cox, M. Niazi, and J. Verner. Empirical Study of Sommerville and Sawyer’s Requirements Engineering Practices. *IET Software*, 3(5):339–355, 2009.
- [Coc00] A. Cockburn. *Writing Effective Use Cases*. Number ISBN-13: 978-0201702255. Addison-Wesley Longman Publishing Co., Inc., 2000.
- [Cru09] D. Cruz. *POSAAM: Eine Methode zu mehr Systematik und Expertenunabhängigkeit in der qualitativen Architekturbewertung*. PhD thesis, Technische Universität München, 2009.
- [CW97] J.E. Conklin and W. Weil. Wicked Problems: Naming the Pain in Organisations. White Paper N/A, Group Decision Support Systems, June 1997.
- [Dam05] W. Damm. Controlling Speculative Design Processes using Rich Component Models. In *Proceedings of the 5th International Conference on Application of Concurrency to System Design (ACSD’05)*, pages 118–119. IEEE Computer Society, 2005.
- [Dav90] A.M. Davis. *Software Requirements: Analysis and Specification*. Number ISBN-13: 978-0138246730. Prentice Hall Press, Upper Saddle River, NJ, USA, 1990.
- [Dav93] A.M. Davis. *Software Requirements: Objects, Functions and States*. Number ISBN-13: 978-0138057633. Prentice Hall Press, Upper Saddle River, NJ, USA, 2nd edition, 1993.
- [DC06] D. Damian and J. Chisan. An Empirical Study of the Complex Relationships between Requirements Engineering Processes and other Processes that lead to Payoffs in Productivity, Quality, and Risk Management. *IEEE Transactions on Software Engineering*, 32(7):433–453, 2006.
- [Dei09] F. Deissenboeck. *Continuous Quality Control of Long-Lived Software Systems*. PhD thesis, Technische Universität München, 2009.
- [DeM79] T. DeMarco. *Structured Analysis and System Specification*. Number ISBN-13: 978-0138543808. Prentice Hall International, 1979.
- [DG96] G. Dinkhoff and V. Gruhn. Entwicklung Workflow-Management-gereigneter Software-Systeme. In G. Vossen and J. Becker, editors, *Geschäftsprozessmodellierung und Workflow-Management - Modelle, Methoden, Werkzeuge*, number ISBN-13: 978-3826601248, chapter 23, pages 405–421. International Thomson Publishing Company, Bonn, Albany, 1996.
- [DJLW09] F. Deissenboeck, E. Juergens, K. Lochmann, and S. Wagner. Software Quality Models: Purposes, Usage Scenarios and Requirements. In *Proceedings of the 7th international workshop on Software Quality (WoSQ 09)*, page N/A. IEEE Computer Society Press, 2009.
- [DKK⁺05] J. Doerr, D. Kerkow, T. Koenig, T. Olsson, and T. Suzuki. Non-Functional Requirements in Industry – Three Case Studies Adopting an Experience-based NFR Method. In *Proceedings of the 13th International Conference on Requirements Engineering (RE 05)*, pages 373–382. IEEE Computer Society, 2005.
- [DKOA06] J. Doerr, T. Koenig, T. Olsson, and S. Adam. Das ReqMan Prozessrahmenwerk. Technical Report IESE-Report 141.06/D, Fraunhofer IESE, 2006.

- [DKVKP03] J. Doerr, D. Kerkow, A. Von Knethen, and B. Paech. Eliciting Efficiency Requirements with Use Cases. In *Proceedings of the 9th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ 03)*, pages 23–32, 2003.
- [DP05] A. Dahledt and A. Persson. Requirements Interdependencies: State of the Art and Future Challenges. In A. Aurum and C. Wohlin, editors, *Engineering and Managing Software Requirements*, number ISBN-13: 978-3-540-25043-2, chapter 5, pages 95–116. Springer-Verlag Berlin, 2005.
- [DVM⁺05] W. Damm, A. Votintseva, A. Metzner, B. Josko, T. Peikenkamp, and E. Böde. Boosting Re-Use of Embedded Automotive Applications through Rich Components. In M. Stoelinga, J. Rehof, and H. Hermans, editors, *Proceedings of Foundation of Interface Technologies (FIT05) in Conjunction with the 16th International Conference on Concurrency Theory (CONCUR05)*, Electronic Notes in Theoretical Computer Science, page N/A, 2005.
- [DW99] W. Dröschel and M. Wiemers. *Das V-Modell 97*. Number ISBN-13: 978-3486250862. Oldenbourg Wissenschaftsverlag, 1999.
- [DWP⁺07] D. Deissenboeck, S. Wagner, M. Pizka, S. Teuchert, and J.F. Girard. An Activity-Based Quality Model for Maintainability. In *Proceedings of the 23rd International Conference on Software Maintenance (ICSM 2007)*, pages 184–193. IEEE Computer Society Press, 2007.
- [EEM95] K. El Enam and N.H. Madhavji. A Field Study of Requirements Engineering Practices in Information Systems Development. In *Proceedings of the 2nd IEEE International Symposium on Requirements Engineering*, pages 68–80. IEEE Computer Society, 1995.
- [EHH⁺08] G. Engels, A. Hess, B. Humm, O. Juwig, M. Lohmann, J.P. Richter, M. Voß, and J. Willkomm. *Quasar Enterprise — Anwendungslandschaften serviceorientiert gestalten*. Number ISBN-13: 978-3898645065. Dpunkt Verlag, 2008.
- [ER03] A. Endres and H.D. Rombach. *A Handbook of Software and Systems Engineering: Empirical Observations, Laws and Theories*. Number ISBN-13: 978-0321154200. Addison-Wesley, 2003.
- [Erl05] T. Erl. *Service-oriented Architecture: Concepts, Technology, and Design*, volume ISBN-13: 978-0131858589. Prentice Hall International, Upper Saddle River, NJ, USA, 2005.
- [ES10] G. Engels and S. Sauer. A Meta-Method for Defining Software Engineering Methods. In G. Engels, C. Lewerentz, W. Schäfer, A. Schürr, and B. Westfechtel, editors, *Graph Transformations and Model-Driven Engineering – Essays Dedicated to Manfred Nagl on the Occasion of his 65th Birthday*, volume 5765 of *Lecture Notes in Computer Science*, pages 411–440. Springer-Verlag Berlin, 2010.
- [EW05] C. Ebert and R. Wieringa. Requirements Engineering: Solutions and Trends. In A. Aurum and C. Wohlin, editors, *Engineering and managing software requirements*, number ISBN-13: 978-3642064074, chapter 20, pages 453–476. Springer-Verlag Berlin, 2005.
- [FBB09] R. Foorthuis, S. Brinkkemper, and R. Bos. An Artifact Model for Projects Conforming to Enterprise Architecture. In J. Stirna and A. Persson, editors, *Proceedings of the 1st Working Conference on the Practice of Enterprise Modeling (POEM)*, volume 15, pages 30–46. Springer-Verlag, 2009.

Bibliography

- [FGP04] A. Fleischmann, E. Geisberger, and M. Pister. Herausforderungen für das Requirements Engineering eingebetteter Systeme. Technical Report TUM-I4014, Technische Universität München, 2004.
- [FHKS08] J. Friedrich, U. Hammerschall, M. Kuhrmann, and M. Sihling. *Das V-Modell XT (Informatik im Fokus)*. Number ISBN-13: 978-3540764038. Springer-Verlag Berlin, 2008.
- [FL03] P. Fettke and P. Loos. Classification of Reference Models: A Methodology and its Application. *Information Systems and e-Business Management*, 1(1):35–53, 2003.
- [Fle08] A. Fleischmann. *Modellbasierte Formalisierung von Anforderungen für eingebettete Systeme im Automotive-Bereich*. PhD thesis, Technische Universität München, 2008.
- [FPQ⁺10] X. Franch, C. Palomares, C. Quer, S. Renault, and F. De Lazzer. A Metamodel for Software Requirements Patterns. In R. Wieringa and A. Persson, editors, *Proceedings of the 16th Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 10)*, volume 6182 of *Lecture Notes in Computer Science*, pages 85–90. Springer-Verlag, 2010.
- [Fri06] J. Friedrich. Technische und Semantische Transformation von Vorgehensmodellen. Master’s thesis, Technische Universität München, 2006.
- [FRO03] B. Fitzgerald, N. Russo, and T. O’Kane. Software Development Method Tailoring at Motorola. *Communication of the ACM*, 46(4):64–70, 2003.
- [FSC06] A. Feller, D. Shunk, and T. Callarman. Value Chains Versus Supply Chains. <http://www.ceibs.edu/knowledge/papers/images/20060317/2847.pdf>, 2006.
- [Gar84] D.A. Garvin. What does Product Quality really mean? *MIT Sloan Management Review*, 26(1):25–43, 1984.
- [GBB⁺06] E. Geisberger, M. Broy, B. Berenbach, J. Kazmeier, D. Paulish, and A. Rudorfer. Requirements Engineering Reference Model (REM). Technical Report TUM-I0618, Technische Universität München, 2006.
- [Gei05] E. Geisberger. *Requirements Engineering eingebetteter Systeme – ein interdisziplinärer Modellierungsansatz*. PhD thesis, Technische Universität München, 2005.
- [GF94] O.C.Z. Gotel and C.W. Finkelstein. An Analysis of the Requirements Traceability Problem. In *Proceedings of the 1st International Conference on Requirements Engineering*, pages 94–101. IEEE Computer Society Press, 1994.
- [GHSY97] I. Graham, B. Henderson-Sellers, and H. Younessi. *The Open Process Specification*. Number ISBN-13: 978-0201331332. Addison-Wesley Longman Publishing Co., Inc., 1997.
- [Gli05] M. Glinz. Rethinking the Notion of Non-Functional Requirements. In *Proceedings of the 3rd World Congress for Software Quality (3WCSQ 05)*, volume 2, pages 55–64, 2005.
- [Gli07] M. Glinz. On Non-Functional Requirements. In *Proceedings of the 15th IEEE International Conference on Requirements Engineering (RE 07)*, pages 21–26. IEEE Computer Society, 2007.
- [Gna05] M. Gnatz. *Vom Vorgehensmodell zum Projektplan*. PhD thesis, Technische Universität München, 2005.

- [GPHS06] C. Gonzalez-Perez and B. Henderson-Sellers. A Powertype-based Metamodelling Framework. *Software and Systems Modeling*, 5(1):72–90, 2006.
- [Gro] The Standish Group. Chaos report. <http://www.standishgroup.com/>.
- [Gro06a] The Capgemini Group. Integrated Architecture Framework (IAF), v4.0. <http://www.capgemini.com/iaf>, 2006.
- [Gro06b] The Open Group. The Open Group Architecture Framework (TOGAF). <http://opengroup.org/architecture>, 2006.
- [Gro09a] SOA Manifesto Working Group. SOA Manifesto. <http://www.soa-manifesto.org>, 2009.
- [Gro09b] The Open Group. Soa ontology. Technical Standard C104 / ISBN: 1931624887, The Open Group, 2009.
- [GSCK04] J. Greenfield, K. Short, S. Cook, and S. Kent. *Software Factories*. Number ISBN-13: 978-0471202944. Wiley & Sons, 2004.
- [Gut94] T. Gutzwiller. *Das CC RIM-Referenzmodell für den Entwurf von betrieblichen, transaktionsorientierten Informationssystemen*. PhD thesis, Hochschule St. Gallen für Wirtschafts-, Rechts-und Sozialwissenschaften, 1994.
- [GW06] T. Gorschek and C. Wohlin. Requirements Abstraction Model. *Requirements Engineering*, 11(1):79–101, 2006.
- [Hag04] P.J. Hagan. Guide to the (evolving) enterprise architecture body of knowledge. Technical Report 04-0104, MITRE Corporation, February 2004.
- [Ham08] U. Hammerschall. *Flexible Methodenintegration in anpassbare Vorgehensmodelle*. PhD thesis, Technische Universität München, 2008.
- [Hau06a] P. Haumer. Eclipse Process Framework Composer – Part 1: Key Concepts. <http://www.eclipse.org/>, 2006.
- [Hau06b] P. Haumer. Eclipse Process Framework Composer – Part 2: Authoring Method Content and Processes. <http://www.eclipse.org/>, 2006.
- [HB01] S. Henninger and K. Baumgarten. A Case-based Approach to Tailoring Software Processes. In *Proceedings of the 4th International Conference on Case-based Reasoning (ICCBR)*, pages 249–262. Springer-Verlag, 2001.
- [Hen98] S. Henninger. An Environment for Reusing Software Processes. In *Proceedings of the 5th International Conference on Software Reuse (ICSR 98)*, pages 103–112. IEEE Computer Society, 1998.
- [HFIM06] K. Hikichi, K. Fushida, H. Iida, and K. Matsumoto. A Software Process Tailoring System focusing to Quantitative Management Plans. In J. Münch and M. Vierimaa, editors, *Proceedings of the 7th International Conference on Product-Focused Software Process Improvement (PROFES 06)*, volume 441–446. Springer-Verlag, 2006.
- [HH00] D. Hay and K.A. Healy. Defining Business Rules — What Are They Really? Technical Report number N/A, The Business Rules Group, 2000.
- [Hof79] D.R. Hofstadter. *Gödel, Escher, Bach: An Eternal Golden Braid*. Number ISBN-13: 978-0465026562. Basic Books, New York, 1979.
- [Hos87] W.A. Hosier. Pitfalls and Safeguards in Real-Time Digital Systems with Emphasis on Programming. In *Proceedings of the 9th Interna-*

Bibliography

- tional Conference on Software Engineering (ICSE 87)*, pages 311–327, Los Alamitos, CA, USA, 1987. IEEE Computer Society Press.
- [HP08] A. Herrmann and B. Paech. MOQARE: Misuse-Oriented Quality Requirements Engineering. *Requirements Engineering*, 13(1):73–86, 2008.
- [HT09] B. Hummel and J. Thyssen. Behavioural Specification of Reactive Systems Using Stream-Based I/O Tables. In *Proceedings of the 7th IEEE International Conference on Software Engineering and Formal Methods (SEFM 09)*, pages 137–146. IEEE Computer Society, 2009.
- [HWFP07] C. Hood, S. Wiedemann, S. Fichtinger, and U. Pautz. *Requirements Management: Interface Between Requirements Development and all other Engineering Processes*. Number ISBN-13: 978-3540476894. Springer-Verlag Berlin, 2007.
- [IEE90] IEEE. IEEE Standard Glossary of Software Engineering Terminology – IEEE Std 610.12-1990. Technical Standard Std 610.121990, The Institute of Electrical and Electronics Engineers, Inc., 1990.
- [IEE98] IEEE. IEEE Recommended Practice for Software Requirements Specifications – IEEE Std 830-1998. Technical Standard IEEE Std 830-1998, The Institute of Electrical and Electronics Engineers, Inc., 1998.
- [IEE00] IEEE. Recommended Practice for Architectural Description of Software-Intensive Systems – IEEE-Std-1471-2000. Technical Standard IEEE-Std-1471-2000, The Institute of Electrical and Electronics Engineers, Inc., 2000.
- [IHMFJ09] S. Islam, S. Houmb, D. Mendez Fernandez, and M. Joarder. Offshore-Outsourced Software Development Risk Management Model. In *Proceedings of the 12th IEEE International Conference on Computer and Information Technology (ICCIT 09)*, pages 514–519, 2009.
- [IIB09] IIBA. *A Guide to the Business Analysis Body of Knowledge (BABOK Guide)*. Number ISBN-13: 978-0981129211. International Institute of Business Analysis (IIBA), 2009.
- [Isl09] S. Islam. Software Development Risk Management Model - A Goal Driven Approach. In *Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundation of Software Engineering (ESEC / FSE)*, pages 5–8, New York, NY, USA, 2009. ACM Press.
- [Isl11] S. Islam. *Software Development Risk Management Model - A Goal-Driven Approach*. PhD thesis, Technische Universität München, 2011.
- [ISO03] ISO/IEC. ISO/IEC 9126: Product Quality – Part 1: Quality Model. Technical Standard ISO/IEC 9126-1:2001(E), International Organization for Standardization, 2003.
- [ISO10] ISO/IEC. ISO/IEC FDIS 25010: Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) – System and Software Quality Models. Technical Standard ISO/IEC 25010:2007(E), International Organization for Standardization, 2010.
- [IT99] ITU-T. Formal Description Techniques – Message Sequence Chart (MSC). Technical Report Z.120, International Telecommunication Union (ITU) Standardization Sector, 1999.
- [Jac00] M. Jackson. *Problem Frames: Analyzing and Structuring Software Development Problems*. Number ISBN-13: 978-0201596274. Addison-Wesley Professional, Boston, MA, USA, 2000.

- [JBR99] I Jacobson, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. Number ISBN-13: 978-0201571691. Addison-Wesley Longman Publishing Co., Inc., 1999.
- [JE03] L. Jiang and A. Eberlein. Decision support for requirements engineering process development. In *Proceedings of the Canadian Conference on Electrical and Computer Engineering (CCECE 03)*, volume 2, pages 1359–1362. IEEE Computer Society, 2003.
- [JL05] P. Jönsson and M. Lindvall. Impact Analysis. In A. Aurum and C. Wohlin, editors, *Engineering and Managing Software Requirements*, volume ISBN-13: 978-3642064074, chapter 6, pages 117–142. Springer-Verlag Berlin, 2005.
- [Kat09] C. Katz. Development of an Artifact Model for measuring the Degree of Completion in IT-Projects. Master’s thesis, Technische Universität München, 2009.
- [KBS04] D. Krafzig, K. Banke, and D. Slama. *Enterprise SOA: Service-Oriented Architecture Best Practices (The Coad Series)*, volume ISBN-13: 978-0131465756. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.
- [KCH⁺90] K. Kang, S. Cohen, J. Hess, W. Nowak, and S. Peterson. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University Pittsburgh, PA., 1990.
- [KH08] M. Kuhrmann and U. Hammerschall. Anpassung des V-Modell XT – Leitfaden zur organisationsspezifischen Anpassung des V-Modell XT. Technical Report TUM-I0831, Technische Universität München, 2008.
- [KK03] P. Kroll and P. Kruchten. *The Rational Unified Process made Easy: a Practitioner’s Guide to the RUP*. Number ISBN-13: 978-0321166098. Addison-Wesley Longman Publishing Co., Inc., 2003.
- [KKZB08] N. Koch, A. Knapp, G. Zhang, and H. Baumeister. UML-based Web Engineering: An Approach based on Standards. In G. Rossi, O. Pastor, D. Schwabe, and L. Olsina, editors, *Web Engineering: Modelling and Implementing Web Applications*, number ISBN-13: 978-1-84628-922-4 in Human-Computer Interaction, chapter 7, pages 157–191. Springer-Verlag, 2008.
- [Kna09] O. Knapp. Analyse und Interpretation von ausgewählten Projekt-Kennzahlen aus sd&m-Entwicklungsprojekten. Master’s thesis, Ludwig-Maximilians-Universität München, 2009.
- [Kof05] Leonid Kof. *Text Analysis for Requirements Engineering*. PhD thesis, Technische Universität München, 2005.
- [KP96] B. Kitchenham and S.P. Pfleeger. Software Quality: The Elusive Target. *IEEE Software*, 13(1):12–21, 1996.
- [KPP94] B. Kitchenham, L. Pickard, and S. Pfleeger. Case Studies for Method and Tool Evaluation. *IEEE Software*, 12(4):52–62, 1994.
- [KR97] J. Karlsson and K. Ryan. A Cost-Value Approach for Prioritizing Requirements. *IEEE Software*, 14(5):67–74, 1997.
- [KS96] G. Kotonya and I. Sommerville. Requirements Engineering with Viewpoints. *The IEE Software Engineering Journal*, 11(1):5–18, 1996.
- [KS98] G. Kotonya and I. Sommerville. *Requirements Engineering - Processes and Techniques*. Number ISBN-13: 978-0471972082. John Wiley and Sons, Inc., 1998.

Bibliography

- [KT07] M. Kamata and T. Tamai. How Does Requirements Quality Relate to Project Success or Failure? In *Proceedings of the 15th International Conference on Requirements Engineering*, pages 69–78. IEEE Computer Society, 2007.
- [KTF10] M. Kuhrmann, T. Ternité, and J. Friedrich. *Das V-Modell anpassen – Anpassung und Einführung kompakt für V-Modell XT Prozessingeneure*, volume 1st. Springer-Verlag Berlin, 2010.
- [Kuh07] M. Kuhrmann. Prozessintegration und -anpassung. Technical Report TUM-I0712, Technische Universität München, 2007.
- [Kuh08a] M. Kuhrmann. Automatisches, werkzeugspezifisches Tailoring für das V-Modell XT. In *Proceedings des 15. Workshop der Fachgruppe WIVM der Gesellschaft für Informatik e.V. (GI)*, pages 86–93. Shaker Verlag, 2008.
- [Kuh08b] M. Kuhrmann. *Konstruktion modularer Vorgehensmodelle*. PhD thesis, Technische Universität München, 2008.
- [LK95] P. Loucopoulos and V. Karakostas. *System Requirements Engineering*. Number ISBN-13: 978-0077078430. McGraw-Hill Inc., New York, NY, USA, 1995.
- [LK06] B. List and B. Korherr. An Evaluation of Conceptual Business Process Modelling Languages. In *Proceedings of the 2006 ACM symposium on Applied computing (SAC 06)*, pages 1532–1539. ACM New York, NY, USA, 2006.
- [LSS10] C. Leuxner, W. Sitou, and B. Spanfelner. A Formal Model for Work Flows. In *Proceedings of the 8th International Conference on Software Engineering and Formal Methods (SEFM 2010)*, pages 135–144. IEEE Computer Society, 2010.
- [LT02] T.R. Lindlof and B.C. Taylor. *Qualitative Communication Research Methods*. Number ISBN-13: 978-0761924944. Sage Publications, Inc., Thousand Oaks, Calif, 2nd edition, 2002.
- [Luk00] K. Lukka. The Key Issues of Applying the Constructive Approach to Field Research. In T. Reponen, editor, *Management Expertise for the New Millenium*, A-1, pages 113–128. Publications of the Turku School of Economics and Business Administration, 2000.
- [LWMFB10] M. Luckey, S. Wagner, D. Mendez Fernandez, and A. Baumann. Reusing Security Requirements Using an Extended Quality Model. In *Proceedings of the 6th International Workshop on Software Engineering for Secure Systems (SESS’10) in Conjunction with the 32nd International Conference on Software Engineering (ICSE 2010)*, pages 1–7, New York, NY, USA, 2010. ACM.
- [MFC09] D. Mendez Fernandez and Capgemini Technology Services. Capgemini Requirements Engineering Method: Quasar Requirements V.1.1. Available on demand at Capgemini Technology Service Deutschland (<http://www.de.capgemini.com>), 2009.
- [MFK09] D. Mendez Fernandez and M. Kuhrmann. Artefact-Based Requirements Engineering and its Integration into a Process Framework: A Customisable Model-based Approach for Business Information Systems’ Analysis. Technical Report TUM-I0929, Technische Universität München, 2009.
- [MFLPW11] D. Mendez Fernandez, K. Lochmann, B. Penzenstadler, and S. Wagner. A Case Study on the Application of an Artefact-Based Requirements Engineering Approach. In *Proceedings of the 15th An-*

- nual Conference on Evaluation and Assessment in Software Engineering (EASE 2011)*, pages 104–113. Institution of Engineering and Technology (IET), 2011.
- [MFPKB10] D. Mendez Fernandez, B. Penzenstadler, M. Kuhrmann, and M. Broy. A Meta Model for Artefact-Oriented: Fundamentals and Lessons Learned in Requirements Engineering. In D. Petriu, N. Rouquette, and O. Haugen, editors, *Proceedings of the 13th International Conference on Model Driven Engineering Languages and Systems (Models)*, volume 6395, pages 183–197. Springer-Verlag Berlin Heidelberg, 2010.
 - [MFWL⁺12] D. Mendez Fernandez, S. Wagner, K. Lochmann, A. Baumann, and H. de Carne. Field Study on Requirements Engineering: Investigation of Artefacts, Project Parameters, and Execution Strategies. *Information and Software Technology*, 54(2):162–178, 2012.
 - [MFWLB10] D. Mendez Fernandez, S. Wagner, K. Lochmann, and A. Baumann. Field Study on Requirements Engineering Artefacts and Patterns. In M. Turner and M. Niazi, editors, *Proceedings of the 14th International Conference on Evaluation and Assessment in Software Engineering (EASE 2010)*, page N/A (Online Publication). BCS eWIC, 2010.
 - [NE00] B. Nuseibeh and S. Easterbrook. Requirements Engineering: A Roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 35–46, New York, NY, USA, 2000. ACM.
 - [NSK00] U. Nikula, J. Sajaniemi, and H. Kälviäinen. A State-of-the-practice Survey on Requirements Engineering in Small-and Medium-sized Enterprises. Research Report 951-764-431-0, Telecom Business Research Center Lappeenranta, 2000.
 - [NTR05] A. Ngo-The and G. Ruhe. Decision Support in Requirements Engineering. In A. Aurum and C. Wohlin, editors, *Engineering and Managing Software Requirements*, number ISBN-13: 978-3642064074, chapter 12, pages 267–281. Springer-Verlag Berlin, 2005.
 - [Ode96] J. Odell. A Primer to Method Engineering. In S. Brinkkemper and R. Lyytinen, K. Welke, editors, *Proceedings of the 8th Working Conference on Method Engineering – Principles of Method Construction and Tool Support*, pages 1–7. Chapman & Hall, 1996.
 - [Off08] P. Offermann. SOAM–Eine Methode zur Konzeption betrieblicher Software mit einer Serviceorientierten Architektur. *Wirtschaftsinformatik*, 50(6):461–471, 2008.
 - [oGCO07] Office of Government Commerce (OGC). *The Introduction to the ITIL Service Lifecycle Book*. Number ISBN-13: 978-0113310616. TSO (The Stationery Office), Norwich, UK, 2007.
 - [oGCO09] Office of Government Commerce (OGC). *Managing Successful Projects with PRINCE 2*. Number ISBN-13: 978-0113308910. TSO (The Stationery Office), Norwich, UK, 3rd edition, 2009.
 - [OMG06a] OMG. Business Process Definition Metamodel (BPD). Technical Standard bmi/2006-09-06, Object Management Group, 2006.
 - [OMG06b] OMG. Meta Object Facility (MOF) Core Specification - Version 2.0. Technical Standard formal/06-01-01, Object Management Group (OMG), 2006.
 - [OMG08] OMG. Software and Systems Process Engineering Meta-Model (SPEM) Specification V. 2.0. Technical Standard formal/2008-04-01, Object Management Group, 2008.

Bibliography

- [OMG10a] OMG. Unified Modeling Language (UML): Infrastructure – Version 2.4 / beta. Technical Standard ptc/2010-11-16, Object Management Group, 2010.
- [OMG10b] OMG. Unified Modeling Language (UML): Superstructure – Version 2.4 / beta. Technical Standard ptc/2010-11-14, Object Management Group, 2010.
- [PC86] D. L. Parnas and P. C. Clements. A Rational Design Process: How and Why to fake it. *IEEE Transactions on Software Engineering*, 12(2):251–257, 1986.
- [PEM03] F. Paetsch, A. Eberlein, and F. Maurer. Requirements Engineering and Agile Software Development. In *Proceedings of the 12th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 308–313, Washington, DC, USA, 2003. IEEE Computer Society.
- [Poh07] K. Pohl. *Requirements Engineering. Grundlagen, Prinzipien, Techniken*. Number ISBN-13: 978-3898645508. Dpunkt Verlag, 2007.
- [Poh10] K. Pohl. *Requirements Engineering: Fundamentals, Principles, and Techniques*. Number ISBN-13: 978-3642125775. Springer-Verlag Berlin, 2010.
- [PPLB07] O. Pedreira, M. Piattini, M.R. Luaces, and N.R. Brisaboa. A Systematic Review of Software Process Tailoring. *SIGSOFT Software Engineering Notes*, 32(3):1–6, 2007.
- [PSP09] B. Penzenstadler, E. Sikora, and K. Pohl. A Requirements Reference Model for Model-based Requirements Engineering in the Automotive Domain. In M. Glinz and P. Heymans, editors, *Proceedings of the The 15th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 09)*, volume 5512, pages 212–217. Springer-Verlag Berlin Heidelberg, 2009.
- [PWBea94] M.C. Paulk, C.V. Weber, Curtis B, and Chrissis M.B. et al. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Number ISBN-13: 978-0201546644. Addison-Wesley Professional, 1994.
- [Rat09] D. Ratiu. *Intentional Meaning of Programs*. PhD thesis, Technische Universität München, 2009.
- [RDR03] J. Ralyté, R. Deneckère, and C. Rolland. Towards a Generic Model for Situational Method Engineering. In J. Eder and M. Missikoff, editors, *Proceedings of the 15th International Conference on Advanced Information Systems Engineering (CAiSE 03)*, pages 95–110. Springer-Verlag Berlin Heidelberg, 2003.
- [RH09] P. Runeson and M. Höst. Guidelines for Conducting and Reporting Case Study Research in Software Engineering. *Empirical Software Engineering*, 14(2):131–164, 2009.
- [Rit08] S. Rittmann. *A Methodology for Modeling Usage Behavior of Multi-functional Systems*. PhD thesis, Technische Universität München, 2008.
- [RJ01] B. Ramesh and M. Jarke. Toward Reference Models for Requirements Traceability. *IEEE Transactions on Software Engineering*, 27(1):58–93, 2001.
- [RP00] C. Rolland and N. Prakash. A Multi-Model View of Process Modelling. CREWS Report Series (accessed on 07/29/2010): <ftp://sunsite.informatik.rwth-aachen.de/pub/CREWS/CREWS-99-09.pdf>, 2000.

- [RPA⁺01] B. Regnell, B. Paech, A. Aurum, C. Wohlin, A. Dutoit, and J.N. och Dag. Requirements Mean Decisions!—Research Issues for Understanding and Supporting Decision-Making in Requirements Engineering. In P. Bengtsson, editor, *Proceedings of the 1st Swedish Conference on Software Engineering Research and Practise*, number 2001:10 (Research Report) in SERP, pages 49–52. Bleking Institute of Technology, 2001.
- [RPB99] C. Rolland, N. Prakash, and A. Benjamin. A Multi-Model View of Process Modelling. *Requirements Engineering*, 4(4):169–187, 1999.
- [RR99] S. Robertson and J. Robertson. *Mastering the Requirements Process*. Number ISBN-13: 978-0201360462. Addison-Wesley Longman Publishing Co., Inc., Amsterdam, 1999.
- [RR07] J. Robertson and S. Robertson. Volere Requirements Specification Templates - Edition 11. <http://www.volere.co.uk>, August 2007.
- [RSM95] C. Rolland, C. Souveyet, and M. Moreno. An Approach for Defining Ways-of-Working. *Information Systems*, 20(4):337–359, 1995.
- [Rup07] C. Rupp. *Requirements Engineering und Management: Professionelle, Iterative Anforderungsanalyse für die Praxis*. Number ISBN-13: 978-3446405097. Hanser Fachbuchverlag, 4th edition, 2007.
- [Sca04] R.W. Scapens. Doing Case Study Research. In B. Lee, editor, *The real Life Guide to Accounting Research: A Behind-the-Scenes View of Using Qualitative Research Methods*, number ISBN-13: 978-0080439723, chapter 15, pages 257–279. Elsevier Ltd., 2004.
- [Sch91] A.-W. Scheer. *Architektur Integrierter Informationssysteme – Grundlagen der Modellierung*. Number ISBN-13: 978-3540554011. Springer-Verlag, 2nd edition, 1991.
- [Sch01] B. Schätz. The ODL Operation Definition Language and the AutoFocus/Quest Application Framework AQuA. Technical Report TUM-I0111, Technische Universität München, 2001.
- [Sch02] A.-W. Scheer. *ARIS. Vom Geschäftsprozess zum Anwendungssystem*. Number ISBN-13: 978-3540658238. Springer-Verlag Berlin, 4th edition, 2002.
- [Sch04] J. Schekkerman. *How to Survive in the Jungle of Enterprise Architecture Frameworks: Creating or Choosing an Enterprise Architecture Framework*. Number ISBN-13: 978-1412016070. Trafford Publishing, 2004.
- [Sch08a] B. Schätz. *Model-Based Development of Software Systems: From Models to Tools*. Habilitationsschrift, Technische Universität München, 2008.
- [Sch08b] B. Schätz. Modular Functional Descriptions. *Electronic Notes in Theoretical Computer Science*, 215:23–38, 2008.
- [SFGP05] B. Schätz, A. Fleischmann, E. Geisberger, and M. Pister. Model-Based Requirements Engineering with AutoRAID. In A.B. Cremers, R. Manthey, P. Martini, and V. Steinhage, editors, *Proceedings zum Workshop Modellbasierte Qualitätssicherung (QUAM)*, Lecture Notes in Informatics (LNI), pages 511–516. Bonner Köllen Verlag, 2005.
- [SJ96] A.-W. Scheer and W. Jost. Geschäftsprozessmodellierung innerhalb einer Unternehmensarchitektur. *Geschäftsprozessmodellierung und Workflow-Management – Modelle, Methoden, Werkzeuge*, pages 29–49, 1996.
- [Smi00] L. Smith. Project Clarity Through Stakeholder Analysis. Technical Report 2000-12-01, Software Technology Support Center, 2000.

Bibliography

- [SPHP02] B. Schätz, A. Pretschner, F. Huber, and J. Philipps. Model-Based Development of Embedded Systems. In J.-M. Bruel and Z. Bellahsene, editors, *Proceedings of Advances in Object-Oriented Information Systems*, volume 2426 of *Lecture Notes in Computer Science*, pages 298–311. Springer-Verlag, 2002.
- [SR05] I. Sommerville and J. Ransom. An Empirical Study of Industrial Requirements Engineering Process Assessment and Improvement. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 14(1):85–117, Jan 2005.
- [SS97] I. Sommerville and P. Sawyer. *Requirements Engineering - A Good Practice Guide*. Number ISBN-13: 978-0471974444. John Wiley and Sons, Inc., 1st edition, 1997.
- [SS06] A. Sillitti and G. Succi. Requirements Engineering for Agile Methods. In A. Aurum and C. Wohlin, editors, *Engineering and Managing Software Requirements*, number ISBN-13: 978-3-540-25043-2, chapter 14, pages 309–325. Springer-Verlag, 2006.
- [Sta74] H. Stachowiak. *Allgemeine Modelltheorie*. Number ISBN-13: 978-3211811061. Springer-Verlag Wien, 1974.
- [Thu04] V. Thurner. *Formal fundierte Modellierung von Geschäftsprozessen*. PhD thesis, Technische Universität München, April 2004.
- [THV97] A.H.M. Ter Hofstede and T. Verhoef. On the Feasibility of Situational Method Engineering* 1. *Information Systems*, 22(6/7):401–422, Jan 1997.
- [TK09] T. Ternité and M. Kuhrmann. Das V-Modell XT 1.3. Metamodell. Technical Report TUM-I0905, Technische Universität München, 2009.
- [vL03] A. van Lamsweerde. From System Goals to Software Architecture. In M. Bernardo and P. Inverardi, editors, *Formal Methods for Software Architecture: 3rd International School on Formal Methods for the Design of Computer, Communication and Software Systems: Software Architectures, SFM 2003*, volume 2804 of *Lecture Notes in Computer Science*, pages 25–43, Bertinoro, Italy, 2003. Springer-Verlag.
- [vL04] A. van Lamsweerde. Goal-Oriented Requirements Engineering: A Roundtrip from Research to Practice. In M. Glinz, editor, *Proceedings of the 12th International Conference on Requirements Engineering*, pages 4–7. IEEE Computer Society, 2004.
- [vL08] A. van Lamsweerde. Requirements Engineering: From Craft to Discipline. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundation of Software Engineering, SIGSOFT '08/FSE-16*, pages 238–249, New York, NY, USA, 2008. ACM.
- [vL09] A. van Lamsweerde. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Number ISBN-13: 978-0470012703. Wiley & Sons, 2009.
- [VMX] V-Modell XT. Definition and Documentation on the Web. <http://www.v-modell-xt.de> (accessed on 7/01/2010).
- [Wag07] S. Wagner. *Cost-Optimisation of Analytical Software Quality Assurance*. PhD thesis, Technische Universität München, 2007.
- [WDW08] S. Wagner, F. Deissenboeck, and S. Winter. Managing Quality Requirements using Activity-Based Quality Models. In *Proceedings of the 6th International Workshop on Software Quality (WoSQ 08)*, pages 29–34, New York, NY, USA, 2008. ACM.

- [WGWP02] J. Ward, P.M. Griffiths, P. Whitmore, and J. Peppard. *Strategic Planning for Information Systems*. Number ISBN-13: 978-0470841471. Wiley & Sons, 3rd edition, 2002.
- [Wie03] K. Wiegers. *Software Requirements*. Number ISBN-13: 978-0735618794. Microsoft Press, Redmond, WA, USA, 2nd edition, 2003.
- [WK92] R.J. Welke and K. Kumar. Method Engineering: A Proposal for Situation-specific Methodology Construction. In S. Cotterman, editor, *System Analysis and Design: A Research Agenda*, John Wiley and Sons, pages 257–269, Chichester, 1992. Wil.
- [WMFIL09] S. Wagner, D. Mendez Fernandez, S. Islam, and K. Lochmann. A security requirements approach for web systems. In M.F. Bertoa, C. Calero, M.A. Moraga, and H. Sharaoui, editors, *Proceedings on the 1st Workshop Quality Assessment in Web (QAW 2009) in Conjunction with the 9th International Conference on Web Engineering (ICWE 09)*, volume 561, page N/A (Online Publication). Sun SITE Central Europe (CEUR), 2009.
- [WRH⁺02] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslen. *Experimentation in Software Engineering: An Introduction*. Number ISBN-13: 978-0792386827. Kluwer Academic Publisher Group, 3300 AH Dordrecht, Netherlands, 2002.
- [Wri91] S. Wright. Requirements Traceability – What? Why? and How? In *Proceedings of the IEE Colloquium on Tools and Techniques for Maintaining Traceability During Design*, page N/A, 1991.
- [WS06] R. Winter and J. Schelp. Reference Modeling and Method Construction: A Design Science Perspective. In *Proceedings of the 21st ACM Symposium on Applied Computing (SAC 06)*, pages 1561–1562. ACM New York, 2006.
- [YMB01] I. Yoon, S. Min, and D. Bae. Tailoring and Verifying Software Process. In *Proceedings of the 8th Asia-Pacific Software Engineering Conference (APSEC)*, pages 194–202. IEEE Computer Society, 2001.
- [Zac87] J.A Zachman. A Framework for Information Systems Architecture. Zachmann Institute for Framework Advancement: <http://www.zifa.com/>, 1987.
- [Zav97] P. Zave. Classification of Research Efforts in Requirements Engineering. *ACM Computing Surveys (CSUR)*, 29(4):315–321, 1997.
- [ZC05] D. Zowghi and C. Coulin. Requirements Elicitation: A Survey of Techniques, Approaches, and Tools. In A. Aurum and C. Wohlin, editors, *Engineering and Managing Software Requirements*, number ISBN-13: 978-3642064074, chapter 2, pages 19–46. Springer-Verlag Berlin, 2005.
- [ZD00] C. Zopounidis and M. Doumpos. PREFDIS: a multicriteria decision support system for sorting decision problems. *Computers & Operations Research*, 27(7):779–797, 2000.
- [ZYCJ06] T. Zhang, S. Ying, S. Cao, and X. Jia. A Modeling Framework for Service-oriented Architecture. In *Proceedings of the 6th International Conference on Quality Software (QSIC 2006)*, pages 219–226. IEEE Computer Society, 2006.