TUM

INSTITUT FÜR INFORMATIK

Artefact-based Requirements Engineering and its Integration into a Process Framework

Daniel Méndez Fernández, Marco Kuhrmann

TUM-I0929
November 09

TECHNISCHE UNIVERSITÄT MÜNCHEN

TUM-INFO-11-I0929-0/1.-FI

Druck: Institut für Informatik der
Technischen Universität München

Artefact-based Requirements Engineering and its Integration into a Process Framework

A Customisable Model-based Approach for Business Information Systems' Analysis

Daniel Méndez Fernández, Marco Kuhrmann

Technische Universität München
Institut für Informatik
Software & Systems Engineering
Boltzmannstr. 3, 85748 Garching, Germany
(mendezfe, kuhrmann)@in.tum.de

Abstract

Requirements engineering (RE) builds a crucial part in software evolution. Nowadays, industries are more then ever facing the problem that the RE process is highly volatile due to the closeness to customer's capabilities, to used process models and to produced specifications. The missing integration of RE into the development life cycle and the missing support of project-specific influences complicate the use of available RE approaches as a standardised, company-specific RE process that fits to the needs of individual (variable) project environments.

In this report we provide an artefact-based RE reference model for business information systems. It serves as an orientation for producing precise specification documents being conformant to the reference model. Based on this reference model, we define a mechanism for a systematic and transparent customisation of the reference approach. The customisation mechanism defines how to customise the reference model at organisational environments concerning process-integration. This process integration is performed with the V-Modell XT, the standard development process for IT-projects within the German government. Furthermore we show how to customise it at project-levels according to variable project influences.

We provide therefore a RE model that is highly customisable and deeply integrated into the development life cycle and prepare its application in real life projects. We lay the foundation for a systematic and integrated RE that fits to the needs variable project environments.

Keywords: Sofware Engineering, Meta Modelling, Artefact Orientation, Software Development Process, Requirements Engineering, Business Information Systems, V-Modell XT

CR-Classification: D.2

Acknowledgements

The work at hand arises from a research cooperation between the Technische Universität München (TUM) and Capgemini sd&m, a major German software development company. The approach for business information systems' analysis has been developed based on experiences of real-life projects regarding analyses and development of custom software in the application domain of business information systems.

This report, however, represents academic and industrial experiences and best practices resulting from the development and the integration of the approach into organisational environment of Capgmini sd&m (see also the results described in [MFsA09]). In this context it is a pleasure to gratefully acknowledge the discussions and fruitful remarks of the colleagues of the TUM and the employees of Capgemini sd&m that supported the development.

Contents

1. Introduction

Requirements Engineering (RE) lays the foundation for successful development projects regarding cost and quality [Bro06][1]. The activities that are performed as part of the RE process aim at the specification of requirements that unambiguously reflect the needs of all relevant stakeholders [FGP04, Gei05]. The precise elaboration of requirements supports subsequent software development processes like architectural design, but also project management processes. For example, requirements are the basis for the definition of contractual agreements [HWFP07]. Hence, RE builds the basis for efficient and effective, customer-oriented development projects.

However, the first step for companies towards RE excellence consists in the definition of a RE process at an *organisational level*. This includes the preparation of methods and techniques for a company-wide use. Furthermore, the process must be integrated into the development life cycle. This integration concerns establishing the association of RE-specific results with contents of further processes of the development life cycle and is known as *process integration*.

Once the RE process has been defined and integrated for a company, it has to methodically guide through the elaboration of requirements at *project-level*, thereby within each individual project of a company. The RE process that is executed at the project-level must then support the precise definition of requirements. Preciseness in the definition of requirements includes aspects of structure, syntax and semantics regarding the content of the specification documents. Obviously, the process must include the basic concepts respecting the needs of individual domains of application. Such a systematic RE approach achieves for requirements engineers *awareness* of constructing requirements specifications that are *conformant to domain-specifics* regarding *completeness* and *consistency* of the results[2]. Completeness means that the results are described with all necessary elements, for example that each use case has a unique identifier. Consistency concerns the consistency in the relations between different results. For example, that each use case description needs the explicit and separate description of an actor. Both completeness and consistency within and between the results are the basis for seamless modelling, thus, for *continuity* between the activities of the RE process and also between RE and further development processes.

Still, in practice the definition and the execution of RE processes is often complicated. It begins with a missing awareness of RE at organisational levels neglecting also its integration into the development life cycle. In addition to missing company-specific RE approaches, different project-specific influences make the process highly variable. A consequence is that engineers often act solution-biased in terms of adopting the solution rather then precisely specifying requirements. As there is no systematic reference approach that can be taken as an orientation for producing precise results, the results remain incomplete and inconsistent. This, in

1 In the context of the introduction we use the term *requirements engineering* in the broader sense. In Sec. 1.5 we give an overview on the essentials of the application domain of business information systems and define in Chp. 3.1 at this basis the discipline *business information systems' analysis* as a roof top over aspects of requirements engineering and aspects of business engineering.

2 The term "domain-specific" addresses in the context of this report a situation, characteristic and / or concern that is typically used in a *domain of application* such as in the one of business information systems.

turn, causes a disruption in the development life cycle, as the results being produced by one development activity are the necessary input for the next. A consequence is an observable quality downslide of the process, thus the results of the process and finally the application being developed.

We subsequently discuss in this chapter the reasons and resulting phenomena of solution-orientation in RE. We elaborate the necessity of a systematic RE approach that supports conformity of the results specific for a domain of application within individual projects. It must be customisable to the needs of organisational environments respecting the process integration into the development life cycle and customisable to the needs of individual projects.

1.1. Problem Statement

Although the research area of RE comprises several well understood techniques and approaches, in practice it is often not possible to apply them as part of a systematic process.

Two reasons for this circumstance are

1. the missing integration of RE at organisational levels.
2. the missing support of project-specific influences that make the process at project level highly variable.

Missing Integration of RE at Organisational Levels. One root cause for a complicated appliance of a systematic RE approach in projects is the missing integration of RE into the development life cycle (at organisational level). The definition of a process and its integration into the development life cycle strongly depends in turn on the domain of application. At the example of business information systems existing frameworks, such as [Zac87, Gro06, Gro], arise from decades in which RE was neglected to be an integrated part of the software development life cycle [Boe06]. Even if the philosophy of "IT Business Alignment" concerns the principles of RE in terms of specifying needs towards IT systems in order to achieve some outcome of value to a customer's business (process landscape), in mentioned standards, the results of RE are still not seen as an integrated part of the development life cycle. Hence, on the one hand, RE is recognised to be a vital part of engineering activities for constraining the current or future state of any aspect of the business and underlying business information systems [BAB09]. On the other hand, there is still a gap between the principles of RE and the architecture driven research area of the application domain of business information systems. Section 1.5 gives an overview over the essentials of the application domain and corresponding (enterprise) architecture frameworks.

However, a consequence of current approaches and frameworks neglecting RE is that companies often do not include RE aspects into their process models. They do either define domain-specific methods and techniques for a systematic elaboration of requirements, nor do they integrate the results into the development life cycle.

Missing Support of Variable Influences at Project Level. The missing integration of RE into the development life cycle takes strong influence on the awareness of engineers towards possibilities and necessities in RE for producing precise results. Even if a company defines and integrates a RE process, it still does not necessarily support individual project-specific influences that engineers have to face at the project level. Examples for such influences are: Constraints onto used specification structures and contents based on e.g. existing documentation; De-

gree of innovation of the application; The motivation, the significance, the availability and the degree of technical ability of the stakeholders and their different needs and perspectives onto the application. Compared to (constructive) development processes, the closeness of RE to the influences of customers makes the process 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 influences pervade therefore not only the set-up, but also continuously the performance of the RE processes and used techniques, as done within no other development process. Besides affecting the necessity and possibility of producing specific results, it affects also the way of producing these results.

Hence, the possible and necessary preciseness and quality of the requirements on which the following development processes build on, is mostly determined by project-specific influences. They complicate a standardisation of RE within individual project environments and thereby the awareness of RE at the project-level.

Resulting Unaware Solution-Orientation in RE. As described, one problem already consists in the missing definition and integration of a RE approach for particular domains of application at organisational level. Even if RE-specific methods and techniques are available they do not resists variable environments at the project levels.

The main effect can be observed in the degree of *solution-orientation*. This means "how fast to switch" to design activities, before or instead of refining and approving requirements with customers to a precise and explicit level. Although there is no exact definition of the term "solution-orientation" yet, it addresses the approach in which requirements remain on a high level of abstraction and directly twisted by solution ideas. Further development processes then act according to incomplete and inconsistent requirements – the solution space then dominates the stated problems. Instead, *problem-orientation* is meant to be the approach of a continuous approval process of problem statement, refinement and problem solving, in which requirements are finally analysed and refined according to the customer's needs to precise requirements.

In fact, 46% to 50% of IT development projects act solution-oriented. This tendency was already observed during the late 1970's and extends to nowadays, as shown by several surveys [Boe06, Boe79, Hos87, Com99, SG02, cha]. Solution-orientation is taken into account in some cases on purpose in order to yield a certain process efficiency and in order to avoid a scope and approval creep (often known as "gold plating" [Wie03]). Still, the unaware solution-orientation is mostly enforced by the missing understanding of engineers towards the necessities and possibilities in RE and project-specific influences.

The consequences of solution-orientation in RE are "underspecified", respectively incomplete requirements (specifications). They do not include measurable and traceable contents, therefore causing inconsistencies among the results. This, in turn, causes a disruption in the development life cycle, as the results being produced by RE activities can not be used as an unambiguous input for further development processes. It negatively affects a continuous development process. As the customer goals can not clearly be proven for satisfaction, this increases the risk of delivering IT systems that qualitatively do not meet the expectations of customers, thus do not support their organisations as expected. The applications do not achieve the expected outcome what, in turn, result in cost-intensive additional development activities, expand the time schedules and the costs of a project [BEJ07, cha] and—an aspect that is often not included within actual surveys—it affects the relation to customers, e.g. regarding missed follow-up

projects.

1.2. Objectives

Companies are facing the problem of standardising an efficient and effective RE process that fits at the same time into individual project environments. This includes (1) the definition and integration of a RE reference process at organisational levels (2) which serves as an orientation at project levels, while (3) supporting a systematic and sustainable decision taking according to project-specific influences. Companies are facing the problem of enabling a *balanced problem-orientation* in RE.

A balanced problem-orientation concerns the definition and integration of a RE approach, specific for one domain of application. Supporting the awareness of the results of RE, it aims in addition within individual projects during the set-up and the performance of a RE process at a continuous reflection on project-specific influences and a systematic decision taking. The decisions refer to act either problem-oriented or solution-oriented, where possible and where necessary. Obviously, it must precisely define when requirements are underspecified and the process therefore solution-oriented (enabling controlling of RE efforts). However, during customisation it must ensure at the same time, independent of taken decisions, sustainable consistency between and completeness of the elaborated domain-specific results.

Summarised, in order to enable a balanced problem-orientation, it demands for an approach that:

1. ensures consistency and completeness of the results to enable a seamless modelling and thereby continuity within the process. It has to precisely define the structure, the syntax and semantics of the results, specific for the chosen application domain.
2. can, at an organisational level, be deeply integrated into the development life cycle, i.e. be integrated into process models establishing relations to development-specific and management processes.
3. guides at project-level the reflection and decision making regarding project influences that arise before and during the execution of RE both affecting the construction of the results.

It must achieve a balanced problem-orientation in terms of supporting awareness of RE processes with precise results respecting conformance to a domain-specific reference. This reference must be customisable for organisations and projects.

Although the necessity of a domain-specific RE reference that is based on a customisable paradigm is understood, there is still little guidance on how to achieve it.

1.3. Related Work

The problem of variability in designing a process in early stages of development is recognised. Still, it is not completely understood. The reasons is that the principles of customisation strongly depend on the chosen paradigm.

We subsequently discuss different paradigms of the research area of RE, the area of business engineering further domain-specific areas for approaches and their possibility of customisation. We show how the concept of artefact-orientation, incorporating the philosophy of model-based development, can achieve a balanced problem-orientation.

In general we can divide the approaches and methodologies into two major paradigms: process-based approaches and artefact-based ones. Both are discussed in the following.

1.3.1. Approaches Following a Process-Based Paradigm

Process-based approaches describe development activities and methodologies to elicit, analyse, document and validate requirements. Approaches that identified a process-based paradigm primarily emphasise the way that requirements are produced or used and steps that should be performed in order to achieve a specific outcome. They define how to do something and benefit a comprehensive description of a process.

Process-based approaches, however, can be structured into single techniques and into comprehensive process models. An example for specific techniques in RE is the *non-functional requirements method* by Doerr [DKVKP03, DKK+05] that proposes the use case-based elicitation and documentation of non-functional requirements in general. Doerr et al. cluster furthermore in *ReqMan* [DKOA06] such techniques and best practices, gained from a survey, into a comprehensive process framework. A further example for such comprehensive frameworks is given by Robertson et al. in [RR99].

Hence, the area of RE offers several well-understood process-based approaches, reaching from heavyweight ones, e.g. described by Wiegers in [Wie03] to agile ones, e.g. described in [PEM03, SS06]. The latter emphasise for example the elaboration and implementation of requirements according to a cost-benefit notion [KR97] implicating an efficient and customer-oriented process in terms of value-based software engineering principles, as proposed by Boehm in [Boe03].

Still, these approaches (and possible combinations of them) have in common that they do not resist varying project situations that go beyond cost/benefit-oriented project influences. Furthermore, they do not ensure conformance to domain-specifics. Hence, independent of the concrete process that is chosen according to an organisational culture, they cannot handle various problems at the project level such as:

- How can restrictions on the used process models be handled?
- How is the process set-up based on specifications that are given by customers?
- What requirements are used in what detail by further development processes. For example, as a basis for performing pricing activities or for defining contracts?

The exclusive description of processes and activities does neither allow any cross-entry into development activities, nor does it support the definition of decisions that have to be made during RE activities. Approaches that follow the process-based paradigm do not offer a structured framework, in which one has the ability of individually acting at the project-level according to varying influences.

These approaches can be – if at all – customised before executing them, i.e. by pruning a generic process model and defining a specific sequence of activities that have to be performed, thus by choosing selected techniques (methods). One example for such a pruning approach is given by Cambell et al. in [CCR+95]. It describes how to define a strict order of activities according to "process drivers". A domain-specific approach is given by the *business analysis body of knowledge* [BAB09], introducing also planning tasks that adjust the activities according to the "project type", but still remaining on an abstract level without giving guidance on how to perform this adjustment. This kind of guidance, however,

has been elaborated by Jiang et al. in [JE03] proposing an algorithm based on previous work of Zopounidis et al. and Eberlein et al. in order to select processes and techniques from a knowledge base according to a specific set of "criteria" (project influences). Both make valuable contributions by including experiences and best practices. Aurum et al. also define in [AW05], based on the ideas of different project influences, a specific model for decision making concerning the support of decision support systems. They define project influences according to management aspects, as well as operational aspects of RE, but remain on an abstract level of implicated techniques (as a cause of the decisions), such as prototyping, similar to previous work in [AW03] and the work of Rolland et al. in [RSM95]. Furthermore, these approaches that arise from the research area of decision theory and decision support systems (see also [RPA+01]) still do not support a process-integration and a continuous customisation of RE with results that are conformant to domain-specifics.

Likewise to the field of decision theory, the research area of *method engineering*, in particular the one of method construction, aims at the systematic definition of methods at an organisational level (see also Braun et al. [BWHW05]). Brinkkemper introduces this research area as a discipline for engineering information systems development methods with a particular focus on tool support in [Bri96]. The (experience-based) choice of methods for individual project environments is introduced and baptised with the term "situational methods".

However, we claim that decision taking for a balanced problem-orientation—ensuring consistency and completeness among domain-specific results, independent of taken decisions and chosen methods—can only be based on well-defined result structure. The result structure and the progress control of the like is not in the primary scope of the introduced process-based approaches. Therefore, this leads to the necessity of process-agnostic RE approaches following a domain-specific artefact-based paradigm.

1.3.2. Approaches Following an Artefact-Based Paradigm

Artefact-orientation in general tackles the variability in designing a process as it concentrates on the results (artefacts) of RE rather then emphasising a strict order of producing the results. The artefact-based paradigm defines the terminology and the content of the results (artefacts) as the backbone of project execution and therefore abstracts from concrete tools or methods. Hence, it overcomes the shortcomings of process-based approaches.

Still, the paradigm "artefact-orientation" is heavily discussed in software engineering and mostly positioned in the context of RE approaches. The terms *artefacts*, *work products* and *product types* often co-exist in same and similar approaches, such as the referred work products that are described in the *rational unified process (RUP)* [JBR99, KK03].

In the research area of RE, we can observe two tendencies and / or understandings within the paradigm of artefact-orientation in dependency to the concern of related approaches:

1. **Understanding emphasising the structure** and used terminology of specifications documents, and concerning a basis for a process integration.
2. **Understanding emphasising the concepts of an application domain** that concern a precise reference for producing the contents of specification documents that are conformant to this reference.

Understanding Emphasising Structure. Approaches that emphasise the structures of the results are often referred to as product-based approaches. Product-based approaches in general describe the development process based on single product (types) as part of comprehensive product models. Product models describe all product types and their relations being produced or used during development processes. The single product types within the product models are often described as a taxonomy. For instance, all documents being produced during by activities of project management, while each document (product type) includes a chapter structure. Product models, specifically elaborated for RE issues, are the *Volere requirements specification templates* [RR07] and the *IEEE Std. 830-1998 recommended practice for software requirements specifications* [oEEE98] that both define the structure of requirements specifications, indicating to the necessary content. These approaches, however, do either define any dependencies between the contents of the specifications, not do they define the (conceptual) content itself. One step towards this direction has been made with the *requirements engineering reference model (REM)* [GBB+06, BPKR09]. REM defines the structure of requirements specifications with a proposed taxonomy [BPKR09] and informally describes dependencies between the elements of the taxonomy based to proposed refinement principles. The defined customisation approach within REM describes to prune for each project-specific contents, to choose methods for producing the contents and to define a process. Still, it does not provide any guidance for this customisation, for example upon which situations to perform what customisation efforts. The reason is that the content and the underlying principles of REM are described independent of domain-specific fine-grained concepts. Thus, it does not provide a precise reference regarding conformance for produced artefacts respecting completeness and syntactic consistency, thereby it does not provide the necessary basis for an organisational and project-specific customisation:

- What are domain-specific modelling concepts and their precise relations that both define the content and the structure of the artefacts?
- What syntax can be chosen for the artefacts and what are syntactical limitations that ensure consistency?
- What are domain-specific refinement hierarchies for the artefacts and how do they merge into existing architectural layers?
- Based on the refinement hierarchies, what exactly means problem-orientation and to what extent are artefacts documented within such approaches (e.g. when are requirements exactly "underspecified")?
- How can the artefact model be incorporated into a process model in order to ensure progress control?

Understanding Emphasising Concepts of an Application Domain. The mentioned shortcomings lead to the second tendency of interpreting artefact-orientation: to the principles of model-based development in which modelling concepts describe those aspects of a system under consideration dealt with during the development process [SPHP02]. How the models are defined depends on their concerns, e.g. concerning the coupling of different tools for different development aspects like aspects of components and their relation to aspects of data modelling [ER03, SPHP02]. The models usually abstract from domain-specific description techniques by characterising the concepts of this domain, reflected by the elements and their relations found in the description techniques [Sch08]. Hence, according to this view, the models are all situated in a specific application domain. Therefore, they are known as domain-specific modelling concepts, as domain models, or as used in the context of this report as *concept models*. An example for such 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 is given by Berenbach et al. that couple in [BW07] concepts of use case modelling and feature modelling or Zhang et al. that define in [ZYCJ06] concepts for modelling services, both describing blueprints of applied single techniques concerning an integrated tool chain. A more comprehensive view is taken by approaches that emphasise whole application domains, such as for the one of web systems, as described by the *UWE* approach in [KKZB08] or the already mentioned *IAF* [Gro]. A further contribution is given by the Object Management Group (OMG) with the standard *SoaML* [Ber08] for service-oriented architectures and UML-based specifications in general. However, approaches like the mentioned ones make mostly use of UML-based modelling languages using UML profiles and focusing on the model-driven methodology in system design (solution design), not taking into account the results of early stages of development.

Comparison and Combination of Both. Compared to approaches that emphasise structural aspects, model-based ones benefit a domain-specific definition of the content of the results. They enable a seamless modelling and thereby achieve consistency among the results. This consistency is based on a concept model that serves as a domain-specific reference for the elaboration of models at the project-level. These models being produced at the project-level then reflect specification contents and relations that are conformant to the domain-specific reference. However, available approaches exhibit the deficiency of not being customisable at organisational levels and not offering a customisable workflow description. The reason is that the explicit definition of results structure is needed for coupling selected concepts to a process model, what in turn is not in scope of concept models[3].

Still, the principles are comprising to enable such a customisation. This is especially true if taking a combination of both the explicit structure of artefacts including domain-specific contents being expressed by concept models.

1.4. Contribution

In this report, we define an integrated and artefact-based RE approach that is customisable towards a balanced problem-orientation. Upon an understanding of artefact-orientation that combines both the structure and content of artefacts being expressed by domain-specific concept models, we can incorporate techniques and notions for producing consistent and complete results.

The proposed artefact model, however, serves then as a reference model for supporting awareness of producing domain-specific results being conformant to the reference model. Based on this reference model, we define a customisation approach that customises the approach at organisational levels and at project levels. Via the definition of the results structure, the artefacts can be integrated into the development life cycle by defining interfaces to selected processes of RE and further processes of the development life cycle. This process integration of the reference model is performed with the V-Modell XT [FHKS08], a German standard for software and systems development projects. Furthermore, artefacts build the necessary basis that can be systematically customised also at project-levels, while still ensuring results that are conformant to domain-specifics. This will be shown by allocating project influences to the artefacts that guide through the elaboration

3 "What entities of the concept model have to be constructed within the activity *Specify Use Case*? What entities of the concept model belong to the use case descriptions within the requirements specification?"

of those artefacts. We define an approach that systematically builds up the process while still achieving consistency (and thus continuity).

We lay therefore the foundation for a systematic and integrated RE that fits into individual (variable) project environments. Figure 1.1 gives an overview on the contributions that achieve the objectives, stated in Sect. 1.2.

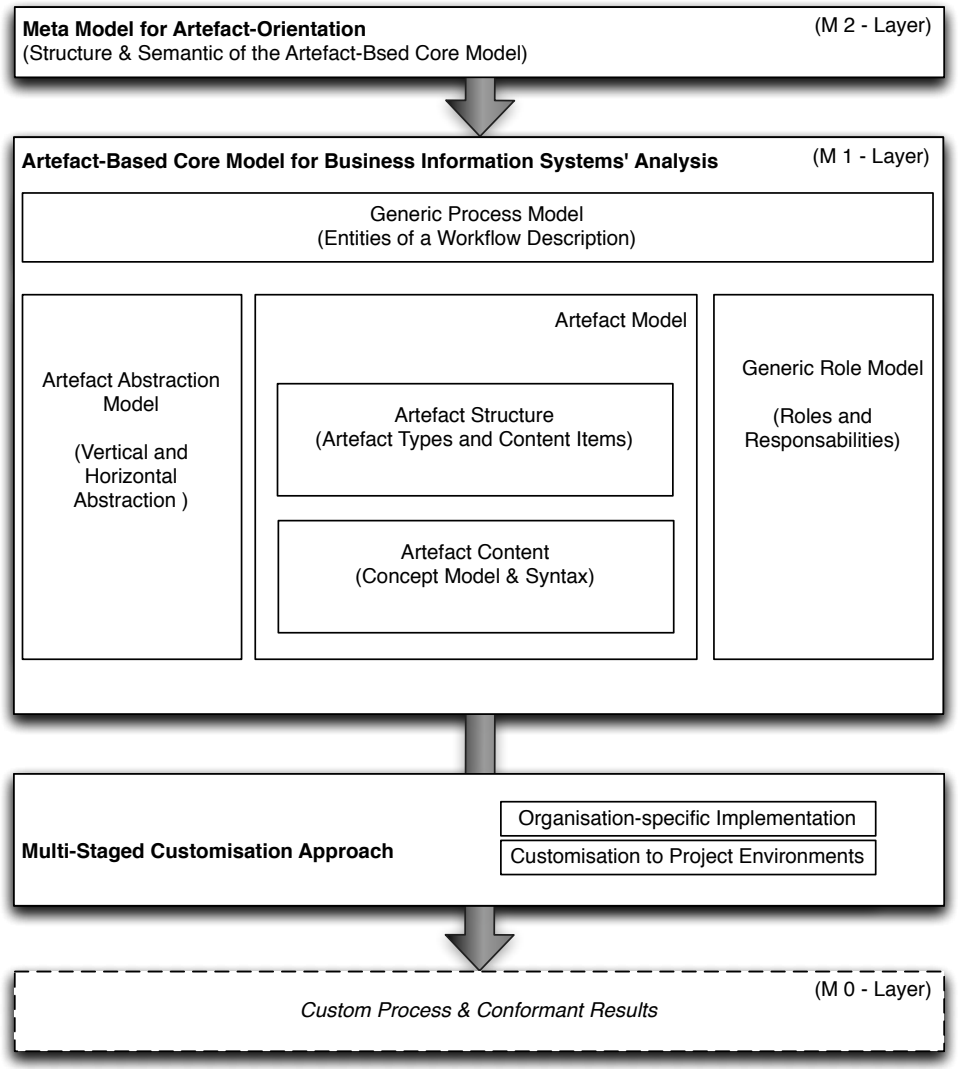


Figure 1.1.: Overview on the Contributions of the Report

In general, we distinguish according to the OMG between three abstractions: a meta model, the artefact-based core model for business information systems' analysis and finally the customisation approach.

Meta Model. A meta model defines structure and semantics of the artefact-based paradigm. A precise structure and semantics is necessary for process integration and customisation. Referring to the terminology of the OMG, this meta model is situated at M2.

Artefact-Based Core Model for Business Information Systems' Analysis. Referring again to the OMG, this model is situated at M1. It contains concrete elements

of the artefact-based infrastructure that serve as a reference model for individual projects. It consists of:

- an *artefact model*: The artefact model is defined according to the necessity of a combination of both structure and content of specification documents. The artefact model defines therefore a (semi-formal) abstraction of structure and contents of specification documents. The structure addresses the aspects of hierarchically ordered documents being produced in terms of a taxonomy in which single content items serve as containers for the concepts, respectively concept models. The concept model (the artefacts' content) defines those aspects of a system under consideration dealt with during the development process [SPHP02] and reflected in the elements and their relations found in the description techniques of particular domains of applications, respectively notions [Sch08]. Both structure and content serve as a reference for the construction of requirements specifications ensuring seamless modelling of the results. Furthermore, we define for each concept in the model possible notions by analysing syntactical possibilities for representing the artefacts. Finally, we define the interfaces of single artefacts to further processes of the development life cycle. The artefact model then ensures domain-specific awareness of precise results as engineers can use this model as an orientation for producing specifications at the project-level that are conformant to the artefact model respecting consistency and completeness[4].
- an *artefact abstraction model*: The artefact abstraction model defines domain-specific vertical and horizontal levels of abstraction over which artefacts are produced, modified and refined (and / or decomposed) and thereby defines completion levels for the concepts. This benefits progress control during refinement of requirements as it fundamentally enables the detection of when requirements are underspecified, i.e. are incomplete and / or remain at a certain level of abstraction. This is necessary to infer a definition of the term "solution-orientation". Otherwise, the decision on if requirements would be underspecified and the process therefore solution-oriented would remain to the expertise of engineers.
- a *role model* and a *process model*: The role model defines the roles and responsibilities that directly participate in the execution of the process regarding the elaboration of the artefacts or indirectly contribute to this elaboration. The process model finally defines a framework with all the entities that are necessary to define a workflow description. This workflow description includes the definition of methods that are performed in order to produce, modify and use artefacts. We define phases, activities and tasks (respectively methods), and milestones, all workflow entities coupled to elements of the artefact model.

Multi-Staged Customisation Approach At one stage we define the mechanism for the integration of the core model into organisational environments. This process integration is performed with the V-Modell XT and lies in the primary focus of this report. However, we define at further stages the mechanism to set-up and to perform the RE process at a project-level (M0) as part of an experienced-based approach similarly as known from decision support systems. The approach is steered by project-specific parameters that affect the elaboration of selected elements of the artefact-based core model (artefacts' structure, artefacts' content, milestones, ...). This finally enables a systematic RE with a transparent decision taking.

4 The primary focus lies on completeness and consistency on the used concepts, the first concerning the contents of the concepts, the latter concerning the interdependencies between the concepts.

1.5. Scope: RE for Business Information Systems

As introductory described in Sect. 1.1, there is a gap between the research area of business information systems and the common understanding of requirements engineering. We give therefore in this section a brief introduction into the fundamentals of the application domain and related terms. We first describe the main principles and characteristics of business information systems and show how the principle of IT business alignment relate to the philosophy of RE. We conclude with the description of exemplary architecture frameworks. The latter is described from the perspective of "to what extent customer demands are defined", i.e. what extent and how IT is aligned with the business needs.

This serves as a basis for defining in Chp. 3 the discipline *business information system's analysis* as a rooftop over the term *requirements engineering* and the term *business engineering*, being in scope of this report.

1.5.1. Steering Principles of the Application Domain

The domain of application of business information systems defines needs towards a specific system's category and its (operational) environment in which the systems are used. The necessities towards the description of the systems and how they must fit into their environment are reflected by methodologies. These methodologies in turn characterise the domain of application by two steering principles that reflect the basic concepts of the domain:

1. *IT Business Alignment*: How can and must information systems support the business (processes) of a company?
2. *Architecture Models*: How must the business and the underlying systems be described and implemented in order to effectively enable this support?

1.5.1.1. IT Business Alignment

IT business alignment concerns the description of how systems must support a company in its "daily business". The description of systems is comprehensively covered with all related logical, technical and technological aspects by the term "Information Technology" (IT). In particular, the principle defines what functionality systems must offer, how they must be structured and what technologies must be used in order to effectively and efficiently support the business processes of a company. The term *IT business alignment* results thereby from a view that aligns two basic abstraction layers: the real world layers describing the business processes of a company and the IT layers describing the systems that support a company in realising their business processes.

Business. The term *business* addresses the business processes of a company that are necessary to achieve some outcome of value. Business processes are described in a sense that reflects the real world as closely as possible. They describe the activities that are performed, by whom they are performed and what information are interchanged. Furthermore, when describing business processes they do not necessarily involve aspects of any underlying systems but emphasise more economically driven aspects. Further arising terms are the one of a *value chain* and the one of a *supply chain*. A value chain describes the linkage between values, being an outcome of business processes. For instance, the value to a customer that makes use of the (business) service such as when ordering 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 enables an economically driven view onto business processes and a flexible integration of business processes, i.e. integration of suppliers, outsourcing business processes and so on [EHH+08].

Information Systems. The term *information system* addresses those aspects of the IT infrastructure that are involved into the realisation of the business (process landscape) of a company and consequently its supply chains. For instance, all aspects that are related to a travel ordering system, necessary to support the business processes of a travel agency. In the context of this report we use the term *Information Systems (IS)*, *Business Information Systems* and *IT Systems* as synonyms regarding information processing systems being structured in order to (at least partially) support business processes. They are characterised by:

1. a boundary, respectively scope
2. a (functional) behaviour that supports business process logic
3. further (also non-functional) properties

The mentioned terms address 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. The term *application* exclusively covers aspects of software, independent of a particular system's (technical) architecture.

RE as the Alignment of Information Systems with Business. The application domain of business information systems is steered by the main objective of describing information systems and needs towards the like in a sense that optimally supports a certain scope of an economically driven business process landscape. The established approaches aim at "aligning the systems with the business" (and vice-versa) [BCVP06, BAB09]. This alignment covers in the nearer sense the philosophy of *requirements engineering* that aims at describing a problem space as comprehensively as possible. RE comprises the iterative and systematic approach of defining a requirements specification aligned to the elementary needs of all relevant stakeholders [Wie03, FGP04, BPKR09]. The term requirement covers in literature business processes and goals (often referred to as business requirements) as well as resulting needs towards the systems [DKOA06, Wie03, RJ01, RR99]. However, as described in Sect. 1.1, RE aspects are still not integrated into the research area of business information systems and vice-versa. Methods and techniques for a systematic elaboration of requirements are still missing in approaches for business information systems. On the other hand, available RE approaches emphasise the description of needs towards systems as a consequence of business needs, but does not include for example methods for exclusively engineering business aspects such as re-designing existing business processes or describing business processes with the intent of outsourcing processes and systems (or single parts of them).

Still, the main idea of RE can be paraphrased with the alignment of the needs towards the *business* with the needs towards underlying *information systems*. For the description of such needs, we make use of architecture models.

1.5.1.2. Architecture Models

Architecture models offer methods to describe the needs of customers towards business (processes) and systems. They guide the elaboration of an architecture in

a standardised way. Still, the term *architecture* is heavily used in the research area of business information systems. We describe therefore in the following two use cases for the term. The first one exclusively addresses the architecture of a single system. The second one uses the term in a broader sense, involving also the business into an architectural perspective and the structuring of systems architectures according to a business architecture. Furthermore, we briefly describe regarding the latter use case two different concepts that concern the alignment of both the systems architecture and the business architecture: enterprise application integration and service-oriented architecrures.

We aim at enabling a common understanding of the essential principles and illustrate the two use cases according to the approach *Quasar Enterprise* [EHH+08]. We briefly discuss the use cases from the perspective of IT business alignment to lay a basis for the description of existing architecture frameworks (and the definition of the discipline business information systems' analysis).

1.5.1.2.1 System Architecture Models

System architecture models describe the basic concepts of single system (and software) architectures. The concepts are reflected by the structure of an architecture using components, the relations in-between the components and the relation to a system's environment, and finally, observable characteristics of the systems [oEEE00, Cru09] [5]. Regarding the observable behaviour of a system, it is described in terms of usage layers [BFG+08], the highest abstraction of a system. Usage layers include the description of how future users intent to use the system when performing their business processes. Although the structure and the functionality of a systems is described according to the needs of the business processes, the system architecture models do not offer means to describe aspects that go beyond the usage of single systems.

System architecture models emphasise single systems and their structure rather than their relations to other systems and to the comprehensive business process structure of a company. The exclusive description of single systems leads for companies especially in the context of application landscapes to the problem of historically grown monoliths. Each system supports a chosen set of business processes and is described without taking a holistic view of further business processes and their relation to other systems into account. Single systems exhibit as a consequence (over the time) complex dependencies to surrounding systems. Changes within the business process landscape often lead to redundancies regarding functionality and consequently the changes not only affect single systems within the application landscapes. The application landscapes are not flexible regarding changes like the ones that are performed when extending existing business processes. Therefore, system architecture models do not effectively support the description of whole application landscapes.

1.5.1.2.2 Enterprise Architecture Models

Enterprise architecture models (often referred to as organisation architecture models) take a more far reaching description of architectures including also the structure of the business and their alignment with an application landscape. The structure of the application landscape includes often two different perspectives onto the architectures: the information or information system architecture including the description of applications and the technology architecture with networks and platforms

5 Although the system architecture includes the logical and the technical description of an architecture (such as used technologies), they are not described in detail since the system architecture is not in scope of this report.

(i.e. whole systems). However, the structure of the business is performed by means like business units. The structure and dependencies of single systems is chosen according to the dependencies in-between the business units of a company [Sch04]. The structured business processes (and further concepts like goals) are baptised with the term *business architecture* [EHH+08]. Structuring both the business and the systems according to each other facilitates the alignment of IT and business. It especially tackles the problem of redundancies and thereby flexibility regarding changes. The reason is that taking such a holistic view enables low coupling between single systems within the IT infrastructure as it is not triggered by single business processes but by the whole structure of a business process landscape.

We describe two basic approaches for elaborating such a holistic view: enterprise application integration and service-oriented architectures.

Enterprise Application Integration. *Enterprise application integration (EAI)* does not only include aspects of single systems, but structures and arranges application landscapes. EAI especially aims at giving guidance on how to describe and implement the architecture of the whole IT infrastructure.

Although, EAI tackles the problem of coupling single systems and thereby gaining flexibility regarding changes in the business processes, it is a purely technology-driven concept. For instance, EAI products make use of interfaces and adapters that enable the coupling of different systems by the use of specific technologies. Still, the alignment of systems and the business is not efficiently gained by EAI. EAI approaches do not provide means beyond a technical integration in order to describe the interdependencies between the business architecture and the IT infrastructure. EAI emphasises the technical coupling of systems rather than their structuring according to structure of a business process landscape of a company. Consequently, redundancies in the functionality of the systems are not effectively avoided when describing systems to support whole supply chains [EHH+08].

Service-Oriented Architectures. *Service-oriented architectures (SOA)* extend the technical view of EAI approaches by means of structuring the business architecture and describing systems according to this structure. Because there are many controversies about SOA we describe only the basic principles. The philosophy of SOA does not primarily describe any technologies such as web services. Instead, SOA emphasises the alignment of both, the business architecture including comprehensive supply chains and the system architecture from a perspective of an application landscape [TOG09].

In order to enable this alignment, a SOA includes (1) the description of the business architecture, (2) the structuring of the business processes by means of domains, respectively business domains, and services and finally (3) the inference of the structure and the functionality of the application landscape according to the business architecture. Compared to EAI approaches, a SOA extends the business layers and the system layers with an additional one that enables a clear transition between both. The transition is performed by the mentioned services and business domains (see also the artefact model in chapter 3). A service is described in a SOA independent of any technologies. They are used as means to describe how the business processes shall be supported by the application landscape [EHH+08]. In a fourth step, the application landscape is structured into single systems.

Finally, how to describe exactly a SOA, depends on the chosen framework. We subsequently give a brief overview on existing frameworks for business information systems and embed the principles of IT business alignment to lay the foundation of the discipline that is in scope of this report.

1.5.2. Frameworks for Business Information Systems

Frameworks for business information systems offer methods, techniques and guidelines. Most of the frameworks in the domain of application use the term "architecture" according to the understanding of enterprise architecture models and are mostly known as enterprise architecture frameworks (see also the fore going section).

Available frameworks can furthermore be categorised into several areas that are in scope of the frameworks. For instance, approaches for describing (engineering) architectures or approaches for assessing existing architectures. A detailed overview on the frameworks is given by Schekkerman in [Sch04] and by the *Guide to the (Evolution) Enterprise Architecture Body of Knowledge* [Cor04].

However, regarding existing frameworks for describing architectures, we can generally divide them into process-based ones and artefact-based ones. The process-based frameworks describe methodologies in terms of defining what has to be done when describing an architecture. A prominent example for such a framework is given by *The Open Group Architecture Framework (TOGAF)* [Gro06]. Instead, artefact-based architecture frameworks give guidance by illustrating the results that have to be produced and used terminology, thereby abstract from concrete methods (see also Sect. 1.3 that describes different paradigms).

In any case, most of the frameworks are based on early efforts of John Zachman, resulting in the *Zachman Framework* [Zac87]. The framework introduces an abstract taxonomy of the results that have to be produced in order to describe an architecture. These results are in turn embedded into a matrix in which the rows describe different "views" (such as the view of an owner, reflected by the business model) and the columns describe different "focusses" (such focussing on data). This matrix served as a basis for defining abstraction in the elaboration and during the evolution of further frameworks.

One exemplary framework that makes use of such abstraction is a commercial one, the *Integrated Architecture Framework (IAF)* [Gro] being developed by the company Capgemini since the 1990s. It is based on experiences of architects made during real-life projects.

We subsequently describe how abstraction is handled within the frameworks at the example of the IAF. Finally, we define according to the understanding of abstraction to what extent business information systems' analysis is, respectively will be covered.

1.5.2.1. Abstraction within existing Frameworks

Fig. 1.2 illustrates at the example of IAF [Gro] how frameworks make use of abstraction. 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" reflect a comprehensive view onto the business of an organisation. The third area "information system" and the fourth one "technology infrastructure" reflect the IT infrastructure that is meant to support the business, respectively that is aligned with the business (see also Sect. 1.5.1).

Architectural Layers define vertical abstraction across all the four architectures. Vertical abstraction is characterised with the "contextual layer" (*Why?*), the "conceptual layer" (*What?*), the "logical layer" (*How?*) and finally the "physical layer" (*With what?*).

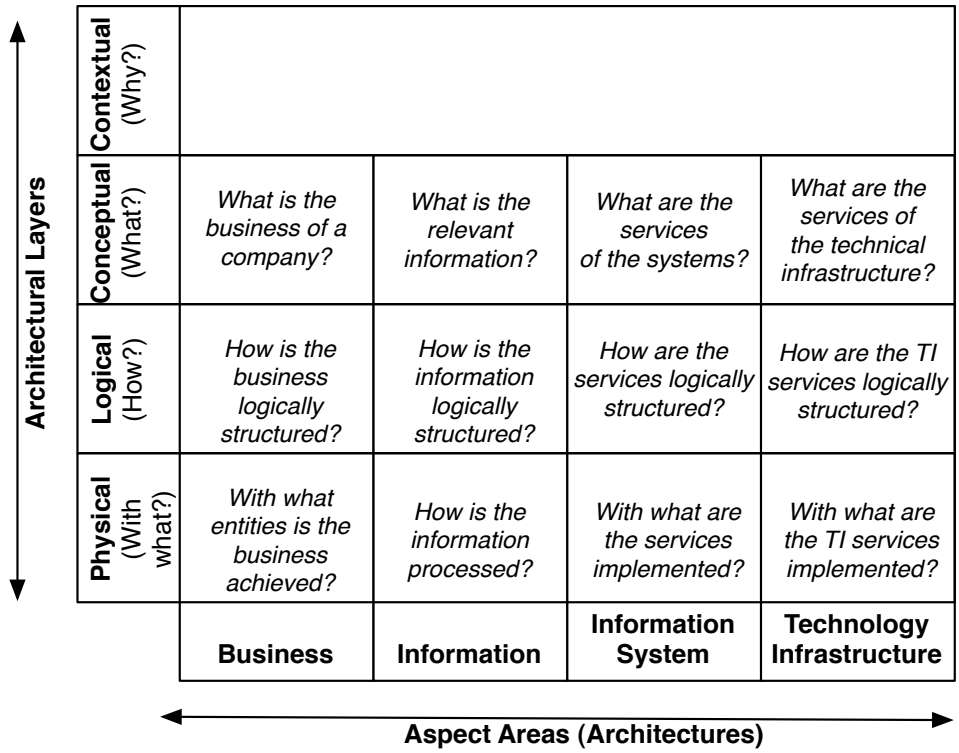


Figure 1.2.: Abstraction at the Example of the Capgemini Integrated Architecture Framework [Gro]

The combination of both dimensions define the backbone of the framework giving with the resulting cells guidance to architects with questions that have to be answered [EHH+08]. Most of existing artefact-based frameworks (like the IAF) embed coarse-grained artefacts into the cells that describe an abstract view onto topics that have to be mentioned within the results (like components and services).

Note that the architectural layers and the aspect areas are taken both in Sect. 3.2 as a basis for the definition of the levels of abstraction (covering vertical abstraction).

Finally, as stated in the introduction in Sect. 1.1, aspects of requirements engineering are in most frameworks reduced to the contextual layers ("why shall something be done?"). As shown in Sect 1.5.1.1, the steering principle of *IT business alignment* paraphrases at the same time essential characteristics of RE.

Hence, we subsequently illustrate under this perspective to what extent IT business alignment and therefore business information systems' analysis over-spans the framework.

1.5.2.2. Business Information Systems' Analysis in the Context of Frameworks

Figure 1.3 illustrates to what extent problem statement, problem solving and the alignment of the both is covered within the frameworks. Dividing the framework into the two major areas meets the claims of the discipline of RE and at the same time the claims of the research area of business information systems.

Note that it is not possible to define any exact borders within the framework, because, first, this depends on chosen methods and approaches for analysis (such as

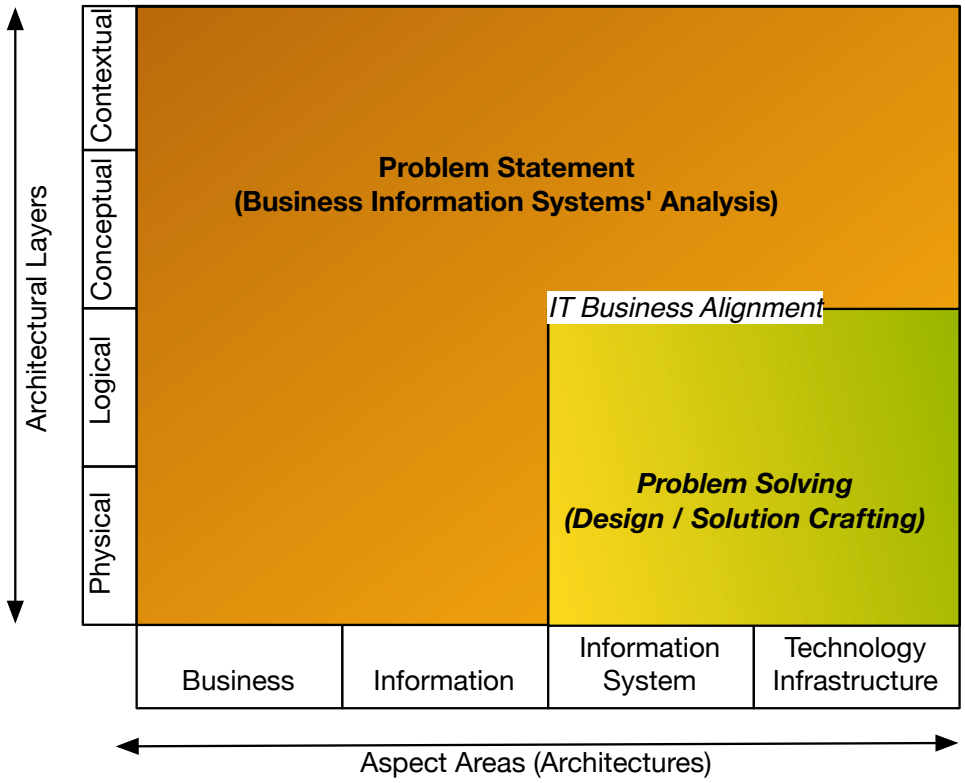


Figure 1.3.: Problem Statement versus Problem Solving

offered by the research area of RE), and second, problem statement and problem solving is performed in iterations, strongly relating to each other.

In the context of this report, we propose therefore also the term *business information systems' analysis* as a rooftop over the principle of describing the needs of the business and resulting needs towards underlying systems. At the same time, we delimit from any aspect that deals with designing an appropriate solution respecting the systems that support the business.

Hence, we claim problem statement to over-span:

1. any aspect that covers the business needs including the business architecture [BCVP06, Wie03, GBB+06, DKOA06]
2. any aspect that covers the needs and constraints towards the systems [BAB09, GW06, Wie03, GBB+06] including also the transition to the (logical) systems architectures in terms of illustrated alignment.

Problem statement covering contextual layers, conceptual layers and further layers that are related to the aspect areas for describing the business is taken therefore as a basis for the definition of the discipline business information systems' analysis in Chp. 3.

1.6. Outline

The remainder of this report is as follows.

Chapter 2 defines in a first step the language for the paradigm artefact orientation and thereby the basis for the definition of the artefact-based core model.

Chapter 3 introduces then the core model for business information systems' analysis. We explicitly define within the artefact model the structure of the specification documents as a taxonomy with respect to existing standards. We enrich the structural view of the documents with a detailed concept model, based on selected description techniques. This concept model defines for selected concepts the concrete details regarding the content of specification documents including the definition of conformant constraints. Based on the artefact model, we define the process model and role model for working out the artefact model and show exemplary methods that can be used. Finally, this chapter defines the interfaces of RE to the development life cycle for the purpose of a clear process integration.

In chapter 4 we define then the multi-staged customisation approach. We introduce the mechanisms for customising the core model of the fore going chapter.

In chapter 4 we perform the organisation-specific implementation, in particular the process integration with the V-Modell XT, before finally concluding in chapter 5 with a final discussion of the results.

Appendix A illustrates the comprehensive artefact model, including the description of exemplary specification structures and a glossary.

2. Meta Model for Artefact-Orientation

We subsequently describe in this chapter the meta model for artefact-orientation. This meta model describes structure and semantics for the paradigm and thereby lays the basis for the concrete artefact-based core model presented in the next chapter 3. We define the following meta model for this paradigm according to the understanding described in the fundamentals in Sect. 1.3.2.

This understanding is coined by the necessity of:

1. Designing a precise model of the results as an outcome of engineering activities while explicitly separating the structure of the results and the content that reflects the basic concepts of a domain of application.
2. Coupling the entities for a workflow description to this model, i.e. designing based on the results a generic process model.
3. Coupling roles to this model in terms of defining responsibilities for the elaboration of the results.
4. Coupling a model for abstraction with the results that allows the characterisation of the results according to defined characteristics. This is necessary for enabling progress control during refinement and enabling the detection of when requirements are underspecified, i.e. are incomplete and / or remain at a certain level of abstraction, and the process therefore solution-oriented.
5. Finally, defining this overall artefact-based infrastructure in a modular way for the purpose of an efficient customisation of single parts, e.g. enabling process integration.

Contents

Figure 2.1 illustrates the meta model of the artefact-based paradigm. The model describes the language for the paradigm and defines single sub-models as major areas of concerns with a specific set of connectors (association classes) in-between these areas. This benefits a modular structure that can be partially customised to organisational needs and instantiated to needs of individual project environments (see also the following chapter 4). Note that the term instantiation is used in the context of this report for reasons of simplicity for both (1) the inference of an instance of the meta model for a specific domain of application (as done in chapter 3) and for (2) the inference of models at the project level based on the domain-specific model.

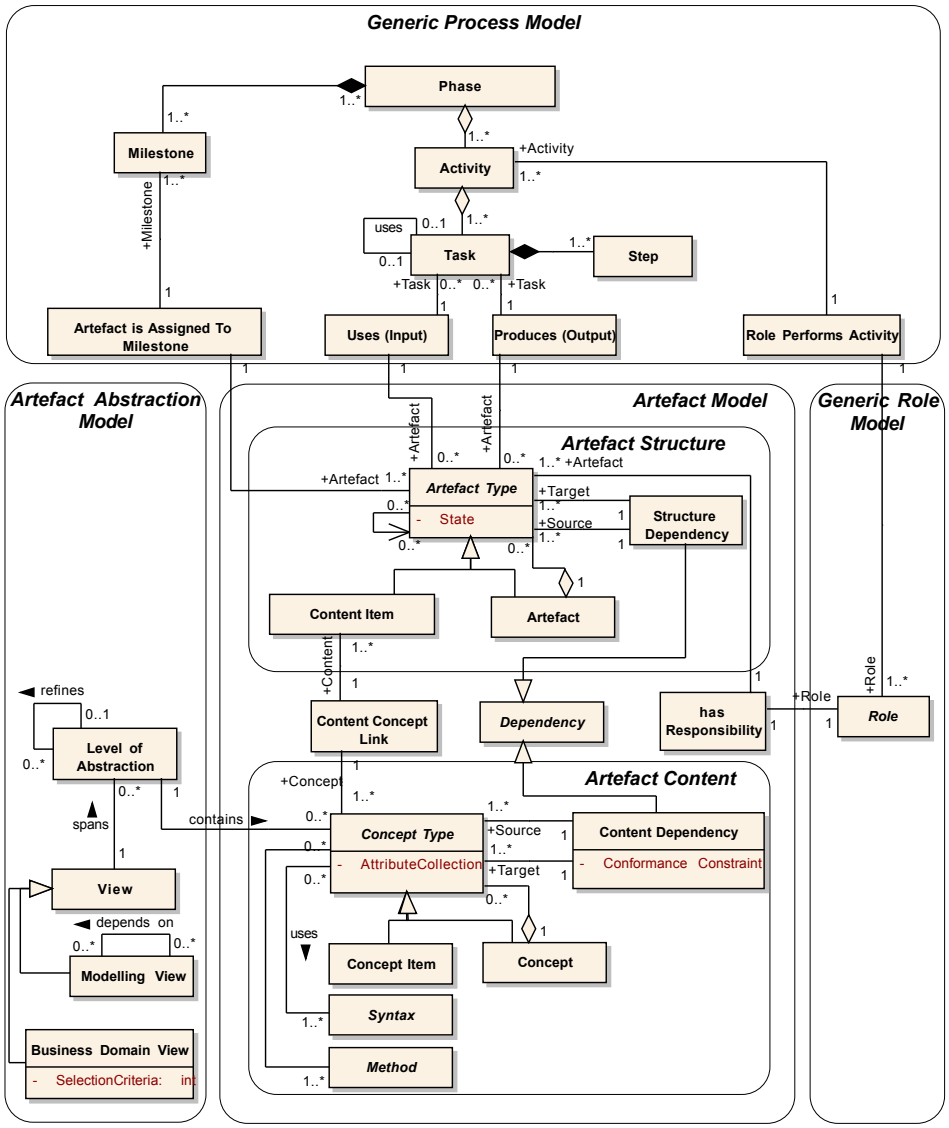


Figure 2.1.: Meta Model of Artefact Orientation

However, the centre of the meta model encompasses the artefact model itself that builds the backbone. It defines a semi-formal abstraction of the structure and the content of the specification documents. It offers a process-agnostic view and benefits the construction of a modular process model, supporting a continuous work-

flow of the expected main results. The artefact abstraction model on the left side of the figure defines the different stages that artefacts pass through during their refinement. The model describes the different levels of abstraction in order to gain progress and maturity control of elaborated artefacts. The upper part of the figure defines the generic process model that describes an abstract sequence of producing the artefacts within a project by coupling activities and milestones to specific parts of the artefact model. The roles that take the responsibility for single artefacts are defined by the generic role model that is depicted on the right side of the figure.

In the following we describe the single parts of the meta model.

2.1. Artefact Model

We explicitly distinguish between artefacts' structure and the artefacts' underlying concepts, i.e. the structure of specification documents and their contents that are embedded into specific containers (content items). The structure is described with a taxonomy reflecting the hierarchical structure of produced specification documents. The included content (the underlying concept model) reflects according to Schätz [Sch08] the concepts of an application domain in terms of precisely defining the used elements and their relations for specific description techniques.

The separation of structure and content benefits:

- with its structural view a clear process integration, as specific parts of the concept model can be coupled to elements of the process model[1]
- with its conceptually defined content and content relations preciseness respecting consistency and completeness of the produced results. It serves as a reference for producing the results in a manner that is conformant to the basic concepts of a domain of application[2].

By building the bridge between principles of model-based development and structural aspects into which the concept model is embedded we support therefore an integrated model that ensures during customisation consistency and completeness among the results. Hence, it also facilitates the possibility of adapting the structure of specifications for example according to constraints. For instance, it facilitates the customisation of the structure of a requirements specification according to the IEEE Std. 830-1998 [oEEE98], without violating the dependencies of the specification's content.

In fact, the separate and explicit definition of the concepts forms a basis for consistency of the content of the specification documents, independent of their structure. Hence, it builds also the basis for an effective progress control and the coupling of different tools.

Based on these differentiating angels from which an artefact is tackled, we define the term *artefact* in the context of this report 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 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 "What elements must be allocated to a chapter that represents a use case model?"

2 "What exact content is needed for describing a use case model and what syntax can be chosen for its representation?"

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 (or task) describes the sequence of steps that is performed in order to make use of a concept.

Example 2.1 illustrates the term of an artefact (and related ones) that is used within this document at the example of a use case.

Example 2.1. *Artefact Types, Artefacts, Concepts and Syntax*

The artefact type *"requirements specification" consists of several* content items, *e.g one including the use case model. Such a content item includes the* concept type *of a use case that in turn consists of a 'name', a 'brief description' and several other attributes that are to be captured when specifying a use case. Furthermore, it groups a set of scenarios and has a specific relation to for example actors that act within these scenarios and that are referred to as concept items. A use case and the included scenarios, however, can be instanciated within a project in different shapes choosing a specific* syntax. *For example using an UML activity diagram in order to illustrate the sequence of interactions (the scenarios) that are encompassed by the use case. Another possibility of representing the use case could be using message sequence charts. The different syntactical possibilities emphasise different aspects of the concepts in dependency to the assertion we want to make during project execution, while the concern of the content item is still the same (to describe users' intents of using an application).*

Note that when referring in the context of this report to artefacts, we usually refer to artefact types ("requirements specification"), rather than to an artefact ("requirements specification 'Travel Ordering System"'). Similarly, when referring to concepts, we refer to concept types ("Use Case") instead of to a concepts ("UML Activity Diagram: Search Hotel").

2.1.1. Artefact Dependencies

The elements within the artefact model can exhibit interdependencies with each other. We distinguish between *structural dependencies* and *content dependencies*.

The structural dependencies are defined by *composition* ("is part of"), as an artefact is decomposed into several content items (i.e. representing chapters within the specification documents). Furthermore, due to the definition of (domain-specific) content dependencies, we also define dependencies between different content items as a consequence. For instance, the content item "Use Case Model" has dependencies to the content item "Actors", as a consequence of the content dependencies of underlying concepts.

The content dependencies are far-reaching and for reasons of complexity not restricted by specific types within the meta model. Still, as a consequence of the relation of single concepts with the levels of abstraction of Sect. 2.2 we can distinguish within all possible instances of the dependencies between:

- *Intra-abstraction dependencies* regarding dependencies between concept of same (vertical) levels of abstraction, such as: "concept A *relates to* concept B".
- *Inter-abstraction dependencies* regarding dependencies between concepts of different (vertical) levels of abstraction, such as: "concept B *refines* concept A".

Finally, as the content dependencies are not typed within the meta model, the domain-specific instances need a definition of conformance constraints.

2.1.2. Content Dependencies respecting Conformance Constraints

Schätz describes in [Sch08] the necessity and possibilities of enriching domain-specific concept models with conformance constraints in order to guide an instantiation of the concept models (M1) to the project level (M0). The definition of conformance constraints supports quality within the models at M0 considered conformant with the reference model at M1. In particular, the application of conformance constraints to the concept models benefit the domain-specific awareness of the results in terms of ensuring:

- *Syntactic Completeness* respecting the presence of specific concepts such as "Each Use Case must exhibit a unique identifier, etc."
- *Syntactic Consistency* respecting the relations between the concepts such as "Each business process must satisfy at least one business goal."

One possibility of defining conformance constraints regarding syntactic completeness and consistency for each concept can be to use logic-based formalisms as done in [Sch01] that enable automated conformance analyses. How to define and ensure the conformance of the models at project level to the concept models, depends on different factors like tool support and is for reasons of simplicity not in scope of this report.

Hint: *Conformance of the Artefact-Based Core Model to the Meta Model*
Note that we do also not define conformance constraints for the meta model considering completeness conditions for domain-specific instances like the artefact-based core model in chapter 3. In fact, the domain-specific instance (M1) is conformant to the meta model (M2) as it is defined with the formulated language of M2.

2.2. Artefact Abstraction Model

The artefact abstraction model defines the domain-specific abstraction that refers to the construction and refinement (and decomposition) of the artefacts. According to Schätz et al. [SPHP02], we distinguish between vertical and horizontal abstractions.

Horizontal abstraction reflects aspects like structure, behaviour or roles and is referred to as *modelling views* (or perspectives onto a system [BFG+08]) [3]. The single views depend on each other, but can be described autonomously, like a behavioural view described with a use case model, while the actors that participate in the scenarios can be worked out in separate views. The separation of such views allows a structured description of systems and benefits an autarchic decision taking in which single aspects can be described without directly having to specify other aspects. However, we distinguish furthermore by modelling views

3 Horizontal abstraction corresponds e.g. in the Zachman framework to the focusses, see also Sect. 1.5.2.

and *business domain views*. Business domains are means to groups specific subsets of artefacts, such as a specific amount of use cases according to defined criteria and will be introduced in section 3.3.

Vertical abstraction describes for each of the views the refinement and / or decomposition of corresponding artefacts. We refer to this abstraction as *levels of abstraction*. The levels of abstraction define domain-specific stages of granularity that artefacts can exhibit depending on the actual scope of refinement, as e.g. depicted in [GW06, RJ01] independent of any domain of application. Referring to abstraction within existing frameworks for the domain of business information systems (see Sect. 1.5.2.1), the levels of abstraction correspond to the architectural layers and the aspect areas.

The stages, however, have specific characteristics that match with the degree of detail of an artefact's underlying concept and finally its concern.

An outcome during a systematic refinement among the levels of abstraction is the gained traceability [RJ01, Pal91] that primarily concerns the (vertical) relations between the artefacts (see also the content item "Traceability Matrix" in Sect. 3.6).

However, the artefact abstraction model is an essential aspect for an efficient process as it benefits:

- a clear definition of refinement hierarchies, enabling progress control independent of a process model such as an incremental one.
- cross entries into the development process, e.g. when setting up a process based on given specifications as their content can be clearly checked according to specific characteristics.
- the definition of completion levels for artefacts and specific stop criteria, e.g. defining precisely when a use case has been elaborated in full and what the exact difference is to a system use case of an internal function specification, still independent of modelling techniques and a chosen syntax.
- the definition of if given specifications imply a solution oriented or a problem oriented approach as one can clearly check if a specification's content is "underspecified" (in terms of if artefacts completely reflect the characteristics of a level of abstraction)

Therefore, the artefact abstraction model builds the basis for evaluating (given) requirements for completion, i.e. precisely defining if they are underspecified and need further refinement by elaborating additional artefacts.

2.3. Generic Process Model

The generic process model defines all the entities of a workflow description that are necessary to define a project-specific process, such as an incremental one. These entities are coupled by the means of association classes to the artefact model and to the generic role model ("what is produced by whom?"). We refer in particular to phases, activities and tasks.

A phase defines a repeatable collection of activities, such as all activities that are performed during the business information systems' analysis.

The activities in turn define major areas of concerns that are performed in order to produce one artefact type. For example, the activity "create business specification" that concerns several all techniques in order to elicit, analyse and validate requirements and finally creates the requirements specification. Obviously, activities are coupled to one or more roles that take the responsibility for producing the related artefact type.

Finally, for each of the activities we define tasks that have an operational character representing the use of techniques. For example, how exactly to elicit functional requirements.

Definition: *Task*

A *task* is a specific sequence of atomic steps that are performed by a role. A task can be initiated by using an output of another task. It reads, modifies or produces one or more concept types with exactly one syntax as an output.

Synonym: Method

While performing a task, we execute in the nearer sense a method in relation to a specific content item, as we make a selection on which concepts an artefact should underlie and which syntax we chose for an appropriate representation for the artefact (see also the research area of method engineering, e.g. introduced in [BWHW05]).

The generic process model offers furthermore the concept of a milestone. Milestones are coupled to selected artefacts and content items. A milestone defines a point in time in which an artefact or content item should be completed. During project execution we decide then on the relation of the milestones to each other and consequently on the order on which the milestones should be reached.

The outcome of the combination of instantiated milestones and workflow entities (phases, activities and tasks) are one specific project execution with one specific process model.

Finally, the model elements such as milestones and roles are underspecified. A precise description has to be delivered by the process model (into which the model is integrated). Of importance are the association classes that state what connection points have to be available. If integrating the meta model, a meta model-based development process model has to be used, which contains at least the elements that are coupled to the association classes such as a milestone element.

2.4. Generic Role Model

The generic role model gives an abstract description of responsibilities that directly participate within or indirectly contribute to the development process. A role can take the responsibility for a specific set of artefacts and performs a set of activities within the process in order to produce or modify the artefacts. This association is reflected by the association classes *Role Performs Activity* and *has Responsilibity*.

However, roles that indirectly participate within the development process may contribute to or depend on the completion of an artefact but have no responsibility for this artefact. By the underspecified definition of a role model, it can be customised to individual needs as the roles can be customised to the role structure of a company and then be allocated at project level to specific individuals within single projects.

3. Artefact-Based Core Model for Business Information Systems' Analysis

We subsequently describe the artefact-based core model for business information systems' analysis. We provide according to the structure of the meta model of the chapter before:

1. an artefact abstraction model in section 3.2 including the definition of (vertical) refinement principles regarding quality aspects and the definition of when artefacts are underspecified (i.e. the process solution-oriented).
2. a complex artefact model in section 3.3. We first give an overview on the envisioned artefact types and describe each in detail afterwards. In Sect. 3.8 we describe how the content of single artefacts interrelates with further development activities like project management activities.
3. a process model in section 3.9.
4. a role model, described in section 3.10.

This chapter describes what is in general produced, how this is done and by whom it is done, independent of any (e.g. economically driven) influences that take effect on the process. How to customise it (1) at organisational level in terms of a process integration and (2) within individual project-environments, e.g. what artefacts not to produce due to specific situations, is described as part of the customisation approach in chapter 4.

Before defining the artefact-based core model we give first an overview on the discipline business information systems' analysis.

Contents

3.1. Business Information System's Analysis

As illustrated chapter 1.5, problem statement can be paraphrased with the steering principle of IT business alignment. We define according to this perspective that the discipline of business information systems' analysis covers any aspect that deals with the description of:

1. the current and the future state of the business (processes),
2. resulting constraints towards envisioned systems.

We use the term *business information systems' analysis (BISA)* instead of RE for several reasons. RE in general emphasises needs towards systems and their development process rather than towards a business. In fact, the (re-)design and description of a business architecture and further concepts that are necessary for the comprehensive description of the like is known under the term *business engineering* [Thu04]. It covers nowadays an own complex discipline with own methods and techniques. Consequently, process models such as the rational unified process [JBR99, KK03], include business engineering (respectively "business modelling") into the development life cycle, besides RE aspects that exclusively concerns the description of demands towards the systems.

Therefore, as depicted in Fig. 3.1, we define the discipline BISA as a comprehensive rooftop over activities that are found in the disciplines of business engineering and requirements engineering.

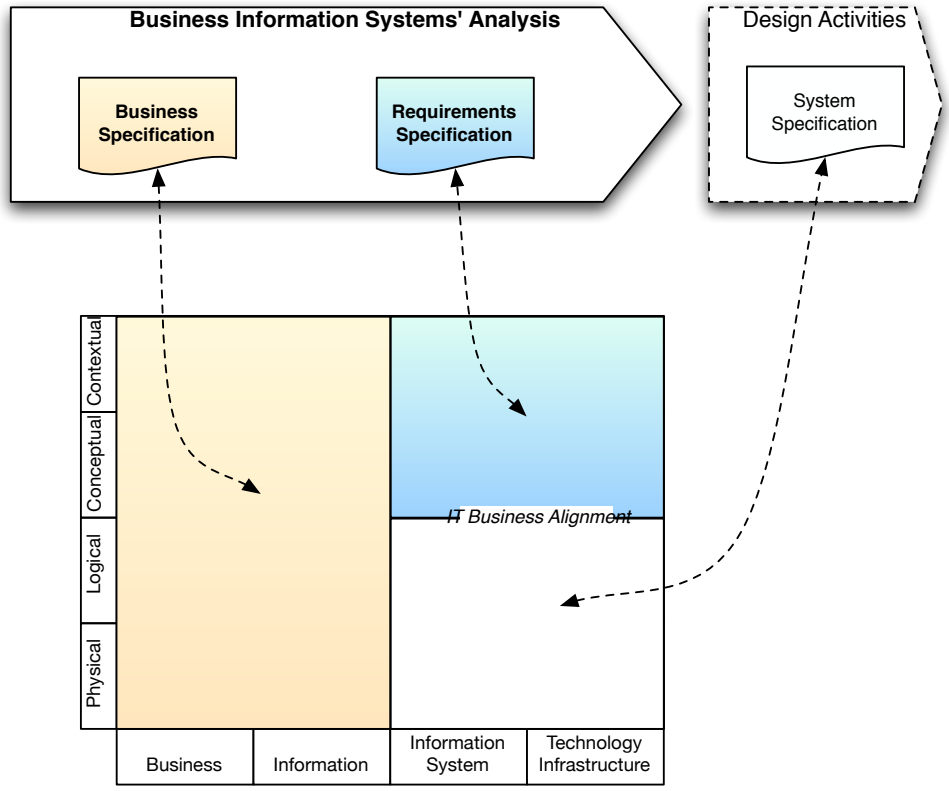


Figure 3.1.: Overview on Business Information Systems' Analysis

BISA covers two basic domains: the business domain (depicted on the left side of the figure) and the domain reflecting resulting demands towards single sys-

tems and their development process (depicted on the upper right side of the figure). The elements that reflect the business (process landscape) are captured by the business specification, while the elements reflecting the demands towards the single systems are captured by the requirements specification (being aligned with the content of the business specification).

According to this view, we define BISA as follows:

Definition: ***Business Information Systems' Analysis***

The discipline *Business Information Systems' Analysis (BISA)* aims at describing the problem space as comprehensively as possible. It comprises activities for an iterative and systematic approach of defining a:

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

In subsequent chapter, we introduce the concrete description artefacts, abstractions and elements of the process model for a business information systems' analysis.

3.2. Artefact Abstraction Model

We first introduce the levels of abstraction to which we refer during analyses and give afterwards a description of the modelling views. Finally, we describe the principles of refining artefacts by embedding them into the artefact abstraction model.

3.2.1. Levels of Abstraction

Figure 3.2 gives an overview on the different levels of abstraction, depicting exemplary questions that symbolise the levels' characteristics.

As described the section before, BISA aims at building the bridge between the needs of a business towards the (future) underlying IT infrastructure in order to achieve some outcome of value. Hence, we divide the levels of abstraction into two major areas that are in scope of this report: one describing the business needs and one representing the needs towards the underlying information systems (supporting the business). The first area is represented in yellow scales including the three initial levels, the second area is represented in blue scales. On the right side of the figure, we sketch the corresponding artefact types. The business specification reflects information about the (idealised) business process landscape of a company and resulting high-level needs and restrictions towards the underlying information system. A requirements specification includes all the requirements towards the information systems under consideration, being aligned with the business specification. With the term *requirement* we refer to requirements towards information systems, their integration and the development process (see also Sect. 3.5).

Obviously, there exist more levels of abstraction that refine the last illustrated one from an architectural perspective (reflecting the system specification). As the focus of this report lies on information systems' analysis, we do not describe them in here in terms of logical and technical architectural layers and resulting system specifications.

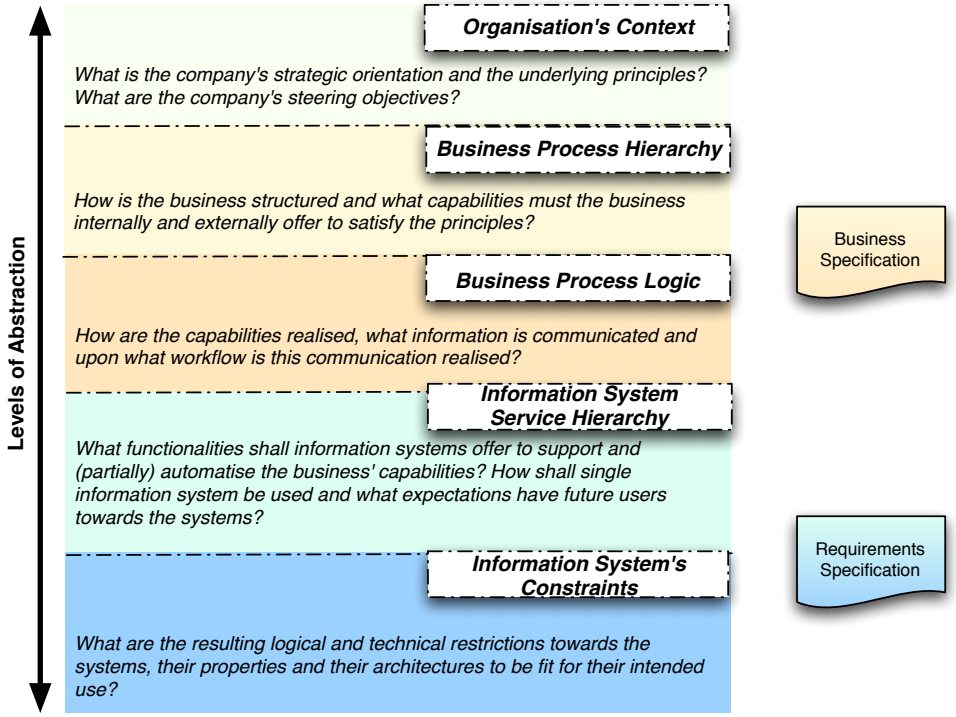


Figure 3.2.: Overview of Requirements Abstraction Levels

During refinement activities, however, we have a satisfy-relation going from top to down. Each level encompasses concepts that have to be satisfied by the concepts of the following levels, independent of the order in which the content is produced.

Organisation's Context. The organisation's context defines the high-level, long-term steering principles and objectives of a company, e.g. the targeted market of a company and how this market shall be incised. The objectives within this layer are the main rationale for the underlying business levels and consequently the IT infrastructure.

Business Process Hierarchy. The business process hierarchy describes the organisation's structure and thereby a taxonomy of the main offered business services that fit into organisation's context. This level does not distinguish between externally offered services and internally offered ones, e.g. services offered to a market and ones offered to internal departments. Within this level, we emphasise the whole business process landscape of an organisation and its structure without a necessary knowledge of its internal realisation.

Business Process Logic. While the business process hierarchy describes structural aspects, this level emphasises the internal realisation of the services, in particular how and by whom they are provided. According to the work of Thurner in [Thu04], this level defines the business process logic taking an operational and internal perspective onto the workflow (the work that is carried out) and concerns all concepts that are necessary to describe that business workflow without an explicit support of particular systems. It includes also the definition of the information that is communicated. For instance, how offered business services are internally realised by a specific set of activities that are performed by business roles

producing and using information objects.

Information System Service Hierarchy. The information system service hierarchy reflects resulting demands towards information systems' functionalities. It includes the description of how particular systems shall support the defined business process logic in achieving the workflow. This is done by defining what business processes shall be at least partially automatised. The concepts that are used to represent this level are known as *user requirements* [Wie03, Coc00, BAB09] that concern the user's expectations towards visible characteristics of the systems as a whole. This level describes therefore how future users shall be supported by the systems when performing their activities. For instance, how the users will interact with the system. The system is seen as a black box that defines the external (user-visible) behaviour without describing any internals of the system, such as the structure of the system by means of (logical) components.

Information System's Constraints. This level of abstraction includes requirements that logically and / or technically constrain a system's architecture. This level is known as the one that encompasses *system requirements* [Wie03] and / or *component requirements* [GW06]. It lays on the boundary to the solution space (design) and concerns the transition to following logical and technical architectural layers. The level includes quantified restrictions and implications towards a system's architecture without necessarily needing knowledge about its internals, thus the system is seen as a grey box (for example restricted with architectural constraints). Requirements at this refinement stage can also arise as a consequence of crafting the architecture, but are still in the primary scope of problem statement.

Solution Space. While the levels depicted by Fig. 3.2 represent the problem space and are in scope of this report ("What has to be done and why?"), the following levels that go beyond the information system's constraints describe the corresponding solution space ("How will it be done?"). They include the system specifications that describe the architecture that is designed as part of the design activities (see also Sect. 1.5.2.2). In particular, following levels include :

1. the logical architectural layer ("How will the system be structured?"), and
2. the physical architectural layer ("With what will the structure be implemented?").

How exactly the layers are represented, depends on the system architecture model that is chosen. One example is given by the one of the Technische Universität München [BFG+08].

Hint: *Levels of Abstraction in Relation to Architectural Frameworks for Business Information Systems*
In Sect. 1.5.2.2 we introduced abstraction in available enterprise architecture frameworks at the example of the IAF [Gro]. The presented levels of abstraction interrelate with the aspect areas and the architectural layers (both stated in combination) as follows:

- The *Organisation's Context* corresponds to the contextual layer
- The *Business Process Hierarchy* corresponds to the conceptual business layer and in parts to the logical business and information layer (that include structural aspects like business units)
- The *Business Process Logic* corresponds to the logical and physical business layer and information layer
- The *Information System Hierarchy* corresponds to the conceptual information system layer

- The *Information System's Constraints* concerns the transition to subsequent layers, i.e. the technology infrastructure and subsequent logical and technical information system layers

Feature- and Goal Levels: Delimiting from existing Approaches

The presented levels of abstraction exhibit differences to existing approaches such as the requirements abstraction model (RAM) [GW06] that is defined independent of any domain of application. In particular, we propose not to describe an explicit level of abstraction for features or for goals as done in RAM. Features relate to a set of requirements and can be stated at different levels of abstraction. Goals are means of making prescriptive statements of intents that are decomposed over several stages. Both the features and the goals are therefore not allocated to an explicit (own) level of abstraction, but relate to selected artefacts that are *refined* over the levels of abstraction (such as the goals being allocated to business processes and business tasks as described in subsequent sections).

3.2.2. Modelling Views

The concepts within the artefact model can be structured according to the following modelling views:

- *Behaviour* defining workflow entities within the envisioned level of abstraction, e.g. described with a business process.
- *Structure* that (logically) groups entities of a workflow description, e.g. described with a component.
- *Environment* describing the environment of envisioned structural aspects, participating and / or motivating a piece of behaviour, such as a role that performs a business process or an external application.
- *Information* describing all the entities that are processed by a piece of behaviour.

3.2.3. Steering Principles for Refinement and Decomposition

We subsequently give a big picture regarding the principles for refinement and decomposition. We first illustrate a combination of both the levels of abstraction and the modelling views, and embed some concepts into the gained raster. In a second step, we use this overview to illustrate in particular how quality aspects and corresponding concepts are related to the levels of abstraction. Finally, we illustrate according to the levels of abstraction the differences between solution-orientation and problem-orientation.

3.2.3.1. Overview on Concepts in Relation to Abstraction

In order to give an introductory overview on the concepts and their relation to the artefact abstraction model, we sketch a combination of both the levels of abstraction and the modelling views. Figure 3.3 illustrates the resulting matrix. According to the characteristics of each cell of this matrix, we embed exemplary concepts (depicted by the white boxes). The figure is reduced to a simplified representation since a detailed description can be taken from the description of the artefact model (see also Sect. 3.3). Furthermore, we do not sketch any process, e.g. by describing a top-down methodology dictating a specific order to make use of the concepts. However, the levels of abstraction build a clear refinement hierarchy that

still symbolises an engagement roadmap. We briefly describe this engagement top down respecting selected concepts.

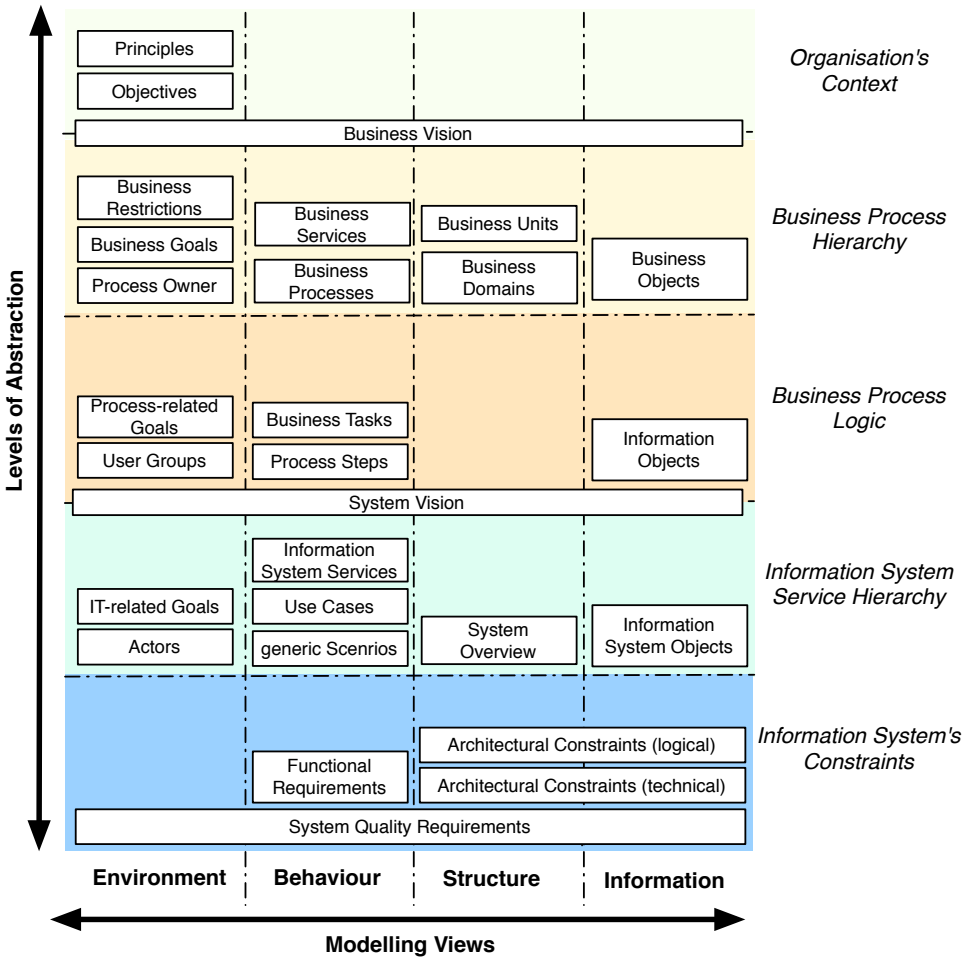


Figure 3.3.: Concepts in Relation to Abstraction

Business Levels. The business process hierarchy describes structural aspects and aims at an administrative perspective. We describe this taxonomy with *business services* and *business processes* and their relation to corresponding *process owners* that take responsibility for the processes. For instance, how a restaurant is structured and who takes the responsibility for specific parts. These parts are known as *business units* and *business domains* that structure the business (processes) into parts that are in scope of the future architecture (e.g. the business domains including the ordering processes that shall be automatised). The internal realisation (the workflow) within the business process logic is described with *business tasks* including fine-grained *process steps*. This workflow builds the basis for defining structure. However, inspired by the approach KAOS (Keep All Objectives Satisfied) in [vL04, vL03] that offers methodology for decomposing goals, we decompose the *business goals* to *process-related goals* for steering the decomposition of the business processes into the direction of the desired workflow. At this level of detail we are therefore able to describe how single *user groups* perform a specific set of atomic process steps in order to achieve one process-related goal. Taking this stop criterion for the business process logic, we can describe for example what steps

a waiter exactly has to perform in a restaurant when ordering and accounting a meal. Similar refinement is done with the processed information. While a *business object* represents an object of reality, such as the "order" itself that the waiter has in his mind, an *information object* describes the content of this order needed when describing the workflow. At this stage of refinement, we are able to analyse how future information systems can automatise the business processes, e.g. what single steps of the waiter shall be supported by the information system in what way.

Information System Levels reflecting the Business Levels. We can make use of similar concepts for describing resulting demands towards the system as done within the business levels. We describe within the information service hierarchy how the systems shall support the business, i.e. how business activities shall be realised by the use of information systems. For instance, in order to achieve a process-related goal like "increase the process lead-time when accounting an order", we can derive *IT-related goals* concerning the quality characteristic "The system shall increase its efficiency". We decide also what information objects shall be reproduced by the systems by specifying *information system objects* (e.g. the order that is processed by a handheld supporting the waiter). Regarding the behavioural transition from the business process logic onto the information system service hierarchy there exist two ways of performing this transition. Either by the means of *information system services* or by describing *use cases*. Both concepts provide means to structure and represent functionality, respectively behaviour of the envisioned system, needed to realise system supported business processes. With use cases, however, we describe a sequence of interactions between *actors* (derived from user groups) and the system in which we analyse how exactly the business tasks shall be supported by the system. Instead, with services we describe a logical representation of the use cases , i.e. functions that the system shall offer. In addition, we make use of *generic scenarios* as means to describe further non-functional expectations towards the system and to describe activities that are performed between several use cases. The last level of abstraction, however, describes demands towards the system that can be classified into detailed quantified requirements. They represent single testable or at least assessable properties of the system. They can be derived from foregoing concepts, e.g. functional requirements from use cases, even if not restricted to it. Note that at this level the concepts can concern several modelling views.

Scoping within the Transitions. Finally, the business levels address whole business process landscape of organisations, while following levels address whole application landscapes. For a systematic and manageable BISA we refer therefore to the means of "scoping". Scoping refers to defining the relevant content subset of one level of abstraction before entering the next. Before specifying the needs of the business, we define the scope within the *business vision*. Before elaborating detailed requirements towards single systems, we define also the scope within the *system vision*. Both visions summarise initial results of following levels of abstraction and align them with the elements that are described within the foregoing level (respectively specification). Furthermore, they define the limitations. For instance, within the system vision we summarise envisioned information system services (resulting from business tasks to be supported) and the borders of the system. Hence, the visions benefit an effective transition between the organisation's context and the following business levels and between the business levels and the levels that concern the systems. The visions abstract from details of following specifications and steer all stakeholders into a common direction before initiating the following analysis activities in full.

3.2.3.2. Refinement and Decomposition of Quality Aspects

System's quality and system quality requirements describe characteristics of systems beyond functionality[1]. Quality aspects need in general a special consideration because they are easily neglected, often difficult to implement and at the same time they are a decisive factor for the success of a project. On the one hand the definition of system quality requirements that sufficiently meet the customer needs is elaborate ("Does this quality result in a value a customer is ready to invest for?"). On the other hand, the requirements remain often "underspecified" (see Sect. 3.2.3.3) so that constructive and analytical development phases are not able to unambiguously interpret, implement and especially control the compliance of the system with the requirement. System quality requirements is in general a topic with many controversies as there exist different views onto the term *quality* with different understandings and therefore a different handling of quality [Gli07, KP96].

Hence, we integrate therefore aspects of quality into the levels of abstraction aiming at a clear integration with the principles of software quality assurance.

There exist related approaches that describe the refinement of quality characteristics in parallel to functional needs handling quality and functionality as two autarkic areas. These approaches describe then the combination of both at their last level of refinement. For instance, by combining system quality requirements with functional analysis models. An example is given by describing use cases in which to state the system quality requirements [DKVKP03, DKK+05].

We take a critical stance towards this autonomous handling of system quality and functional requirements and their late combination during RE because: (1) a clear differentiation between functional requirements and system quality requirements is only possible at their last level of abstraction in which the concepts exhibit different concerns [Gli05, Gli07] and (2) especially think that the cross-cutting nature of quality requirements demand a clear integration of refinement principles into analytical software quality assurance that uses established quality (definition) models as their backbone (see also [DJLW09]).

Referring to such quality definition models that describe "what quality is", we integrate therefore in particular the activity-based quality model of the Technische Universität München that is based in turn on early work of Boehm et al. in [BBK+78] and McCall et al. in [CM78]. The steering theme is that the quality of an information system and therefore system quality requirements are expressed according to the philosophy of: *How good are (business) activities supported by the systems in order to achieve defined (business) goals?*

We emphasise the activities that are performed and supported by systems and define with system quality requirements how specific properties of the system and parts of it must achieve this support. For example by analysing what activities an administrator performs on the system and defining what properties the technical architecture shall exhibit in order to support these maintenance activities. We benefit with this view a cost-benetif notion by enabling the calculation of the costs of performed activities using activity-based costing (ABC) methods. We already showed and evaluated in [WMFIL09, WDW08] the synergy and integration of this model into RE to build the bridge into analytical software quality assurance.

Therefore, as depicted by Fig. 3.4 we perform the refinement of quality aspects by two ways: by (1) a continuous decomposition of goals and objectives over several stages which are used according to Cockburn [Coc00] to steer (2) the decompo-

1 See also Sect. 3.5.1 that introduces the different requirements classes and content item 3.5.3.5 that defines the term *system quality requirements* and its relation to similar terms.

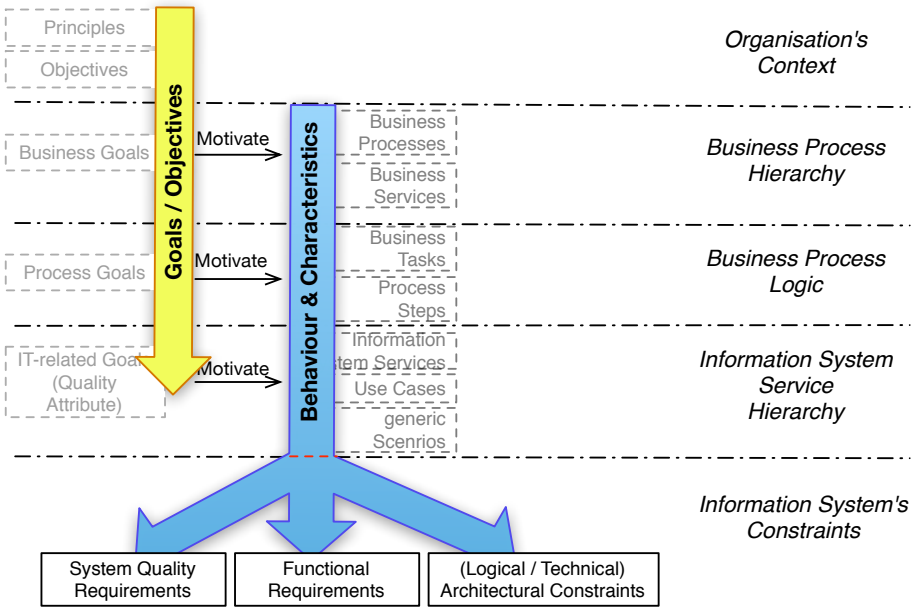


Figure 3.4.: Refinement of Functional and Quality Requirements

sition of behavioural aspects and characteristics of the business into a customer-adequate direction, including structural aspects, internal workflows of the processes and finally how these shall be supported by the systems. The behaviour is in the latter case described with means like scenarios and use cases. They show how specific business activities shall be supported by the system while meeting in particular IT-related goals (quality attributes, like "maintainability"). Note that we refer to scenarios not only to describe functionality. An example is given by describing security requirements in terms of attack scenarios that shall be prevented, i.e. "misuse cases" [HP08], or by describing security requirements in terms of a use case that specifies a login-procedure. For this purpose we introduced also in addition to use cases generic scenarios that illustrates these quality needs. Anyway, this is especially the reason why we delimit from combining quality requirements to e.g. use cases, as scenarios are general means to express behavioural demands towards systems — functional and quality ones.

Hence, at the last level of abstraction, we can derive from mentioned scenarios quantified (or at least assessable) requirements and classify them according to Glinz [Gli05, Gli07] only at this level according to their concerns towards system's details into:

- Functional requirements, describing single detailed actions a system must perform (see also content item 3.5.3.3.2).
- System quality requirements, describing specific metrics and values that constrain the system and its environment in its properties and conditions (see also content item 3.5.3.5).
- Architectural requirements, restricting logically and physically orthogonal to both upper types the system in its architecture — independent of the effects towards the system's quality (see also content item 3.5.3.6).

Although we distinguish more requirements types having organisational nature, we introduce them besides other concepts like business goals within the artefact

model.

3.2.3.3. Solution-Oriented and Problem-Oriented Refinement

Business information systems' analysis is in general an approach of iterative problem statement and solution crafting. It is nearly impossible to completely formulate a problem with requirements with all their details in depth without having continuously progressing an idea of the future system in mind [Wie03, GBB+06, BPKR09]. Hence, the elaboration of a future architecture begins constantly during the elaboration of the requirements, as we continuously reflect with each stated requirement on possibilities of solving the problem. With each design decision that we make, we narrow down the problem space for further requirements [CW97] (see also Sect. 3.9.2 introducing the process model).

Still, as described in the introductory part of this report, a major problem in practice is the performed solution-orientation, based on domain-specific awareness of the artefacts to be produced. This means to switch too fast to solution crafting and leaving requirements on a high level of abstraction. The consequence is the possible exclusion of customer-adequate solution possibilities as engineers then act according to a problem that is not understood in full. The result of this approach are high-level and incomplete artefacts that do not reflect the real needs of customers.

However, the term *solution-orientation* often arises in literature, but mostly related to specific techniques (e.g. in [DKVKP03]), or related to *underspecified requirements* [Poh07]. In fact, it is still not properly defined when requirements (or other artefacts) are underspecified because it is mostly left to the expertise of engineers to decide if statements remain underspecified and the process is therefore solution-oriented. Indeed, the terms cannot be clearly defined because this strongly depends on the concept models of an application domain and their project-specific elaboration respecting (syntactic) completion. Solution-orientation depends of the artefact model's underlying levels of abstraction that allow a rating of single artefacts according to their completion and their allocation to specific levels of abstraction (see also Sect. 2.2).

Therefore, based on the defined levels of abstraction that describe the stages of refinement we are able to decide on if single artefact types and single contents of the artefact types remain underspecified and the resulting approach is therefore solution-oriented. Note that a solution-oriented development process can affect a business specification as well as a requirements specification, because both are used to describe the problem space as part of a business information systems' analysis .

Definition: ***Underspecified Artefacts***

A *content item* is underspecified if it is in a state being not conformant to the corresponding concept type, i.e. *incomplete and / or inconsistent*.

An *artefact type* remains underspecified if the corresponding concept types do not underlie further refinements remaining at the level of:

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

Therefore, we see a single concept as underspecified if it is not conformant to the concept model. An artefact type is underspecified if whole content items are un-

derspecified (or missing). Both aspects take influence on the degree of solution-orientation.

Figure 3.5 illustrates the differences between a problem-orientation and solution-orientation going from left to right.

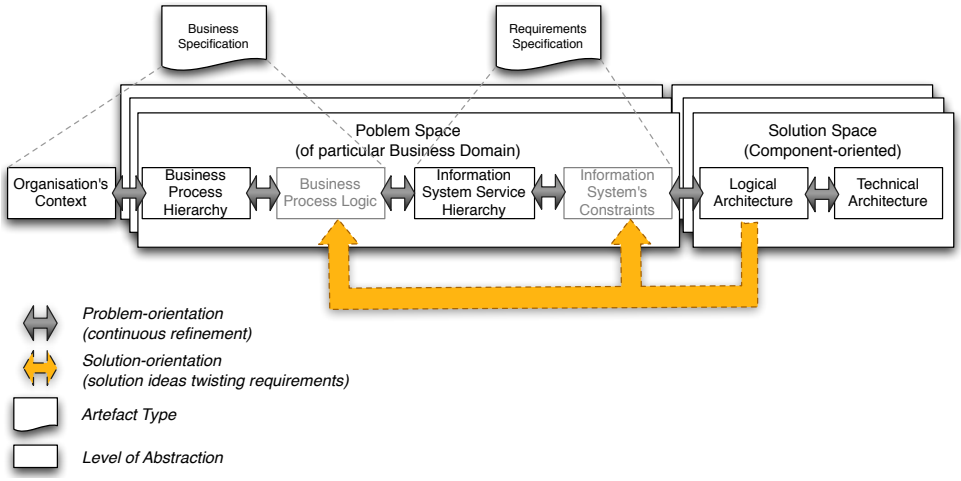


Figure 3.5.: Solution-Oriented versus Problem-Oriented Refinement

In the case of (pure) problem-orientation, 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 — independent of any order in which the concepts are produced. This problem-oriented refinement principle is depicted with the dark arrows. Instead and depicted by orange dotted arrows, solution-orientation describes the approach in which solution ideas dominate the stated problems. In particular, the IT system service hierarchy would be absorbed according to a gained understanding of the logical architecture being 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 result would be discussions and negotiations with customers that are based on an architecture-driven solution at the cost of perhaps missing important information on the workflow of the business processes and missing (quantified) requirements that both are not worked out, but perhaps also with the benefit of a more efficient (development) process.

We can in conclusion define the terms *solution-orientation* and *problem-orientation* in the context of this report as follows:

Definition: ***Solution-Orientation and Problem-Orientation***

Solution-Orientation is the approach in which artefacts remain underspecified. They are directly distorted by solution ideas of following constructive development phases. The concepts that describe the problem space remain incomplete at the level of the business process hierarchy and / or the IT system hierarchy.

Problem-Orientation is the approach of a continuous approval process with a customer in which the artefacts are analysed, refined and quantified in consistency and compliance to the growing logical architecture.

Finally, the degree of the necessary problem-orientation varies from project to project and from business domain to business domain, depending on project parameters that take effect on the construction of single artefacts. For instance, the technical abilities of the stakeholders taking influences on the ability of defining detailed system quality requirements (see also Chp. 4 that describes customisation according to such influences).

3.3. Artefact Model: Overview

The artefact model covers three artefact types: the business specification, the requirements specification and the traceability matrix (see also figure 3.6 illustrating them from a structural perspective).

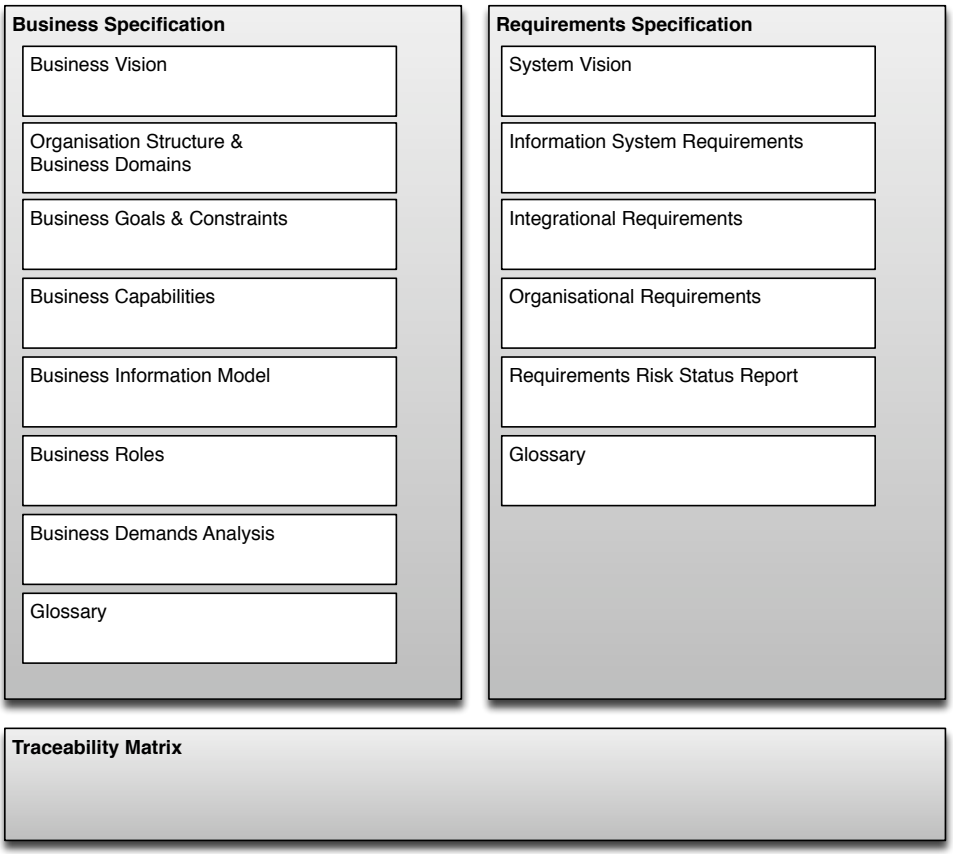


Figure 3.6.: Overview on Artefact Types

The business specification is described in Sect. 3.4, the requirements specification in Sect. 3.5 and the traceability matrix finally in Sect. 3.6. As we classify only those aspects as requirements (types) that directly affect information systems or the development process of the like, we give also in corresponding section an introduction into the different requirements classes and propose important requirements attributes. In Sect. 3.7, we give a brief overview onto a possible (generic) system concept, depicting how the artefacts of the requirements specification interrelate with a system specification.

For reasons of complexity the artefact model is described according to the hierarchical structure of the artefact types as ideally done when constructing the artefact

model during project execution. We guide through the single content items, respectively chapters, of the specifications.

Within each artefact type we first give an overview on its structure and subsequently describe each of the content item according to:

1. the underlying (excerpt of the) *concept model* with a definition of the concepts (in given figures, the content items are coloured in grey scales, while the encompassed concept types are coloured in white)
2. its *content dependencies* considering the relations of the concepts as a basis for consistency conditions (see also section 2.1)
3. corresponding *syntax* regarding possibilities for choosing the notion when constructing the artefacts, while we give wherever reasonable (1) recommendations for a syntax and (2) illustrate an example.

The comprehensive concept model that can be used as a basis for defining for example UML profiles (concerning an integrated tool support for business information systems' analysis) is defined in appendix A.2. Appendix A.1, in turn, illustrates the artefact structures, i.e. ways of organising the specification documents.

3.3.1. Content Dependencies

The meta model in section 2.1 does not demand the definition of specific dependency types between the concepts and most of the dependencies are self-explanatory. Still, in order to facilitate the reading of the illustrated excerpts of the artefact model, we subsequently explain three frequently used dependencies:

- *derives*: If a concept A is derived from a concept B, the content of A can only (at least partially) be defined at the basis of the content of B, implying also an order in the sequence of producing the content (within the process model). Note that if A is derived from B, A automatically satisfies B, but not necessarily vice-versa.
- *satisfies*: If a concept A satisfies a concept B, they make both different statements while the concern of A supports the concern of B. An example would be that the concept "scenario" satisfies the concept "business goal", as a scenario — e.g. as part of a use case — supports in achieving a business goal. Furthermore, if A satisfies B, it must not necessarily have been derived from B. For example, at the project level the scenario "calculate tax" would satisfy the business goal "reduce process lead-time", but the scenario could be derived from a business rules "tax calculation" describing actionable standard procedures within a company.
- *refines*: If a concept A refines a concept B, A is a proper subset of B, while being situated within a lower level of abstraction then B. In addition, B can then be reproduced at the basis of A. However, if A refines itself, it is refined over several levels of abstraction while still referring to same (modelling) concepts.

3.3.2. Artefact Status Model for Progress Control

Each artefact within the model is subject to progress control within a project. For this purpose we define a status model that enables the definition of the current state of an artefact during a project. Figure 3.7 illustrates according to the status model of the V-Modell the possible states of an artefact.

We distinguish subsequent states:

1. An artefact can be *created*.

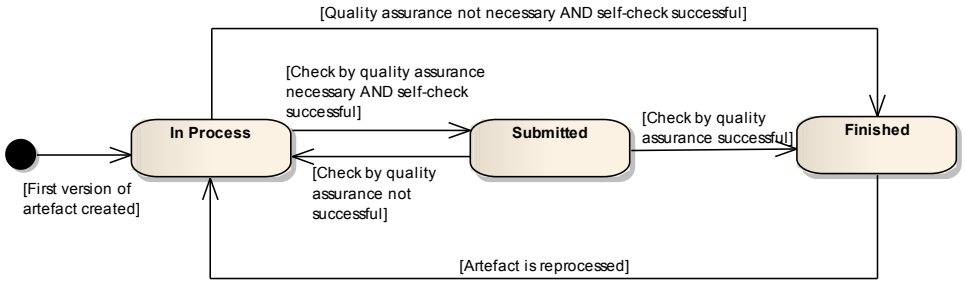


Figure 3.7.: Artefact Status Model

2. An artefact is after its creation *in process*, i.e. being modified during analysis activities.
3. If no (external) quality assurance is assigned, the artefact is after a self-check (of the responsible role) *finished*. With quality assurance we address techniques that are applied from independent persons. For example as it is the case within quality gates that check artefacts with means like inspections. This aspect covers also an acceptance of external parties like the customer itself.
4. If a quality assurance is necessary, the artefact is *submitted*, e.g. proposed to an inspection. The check can be successful or fail. If it fails the artefact is modified again (in process). Otherwise, the artefact is approved and thereby finished.

Note that when finishing an artefact this implies its conformance to the artefact model. It can then be used by further tasks as an input. For example by project management for performing estimations of effort regarding pricing activities.

Finally, the status model can be extended with further states that emphasise states resulting from the quality assurance such as by defining the state "in revision". An example for such a status model can be taken from [Kat09].

3.4. Artefact Type: Business Specification

The business specification formulates all goals, capabilities, restrictions and conditions that affect the business of a customer's organisation and further information that describe the current and future state of the like. Although most of the business needs affect the underlying IT infrastructure, the specification is described without directly involving any aspects of envisioned business information systems. Figure 3.8 gives an overview on the structure of the business specification, i.e. its hierarchically ordered content items that are subsequently defined. We define:

- the *business vision* in content item 3.4.1: "What is the main scope?".
- the *organisation structure and business domains* in content item 3.4.2: "How are the organisations structured?".
- the *business goals and restrictions* in content item 3.4.3: "What are the steering goals and constraining restrictions?".
- the *business capabilities* in content item 3.4.4: "What activities are performed and what services are offered in the envisioned organisation?".
- the *business information model* in content item 3.4.5: "What information is communicated?".

- the *business roles* in content item 3.4.6: "Who participates within the activities and / or takes influence on the definition and performance of the like?".
- the *business demands analysis* in content item 3.4.7: "What are the demands and the benefits that enforce a change in the current state of the business?".
- the *glossary* in content item 3.4.8: "What are the essential domain-specific terms?".

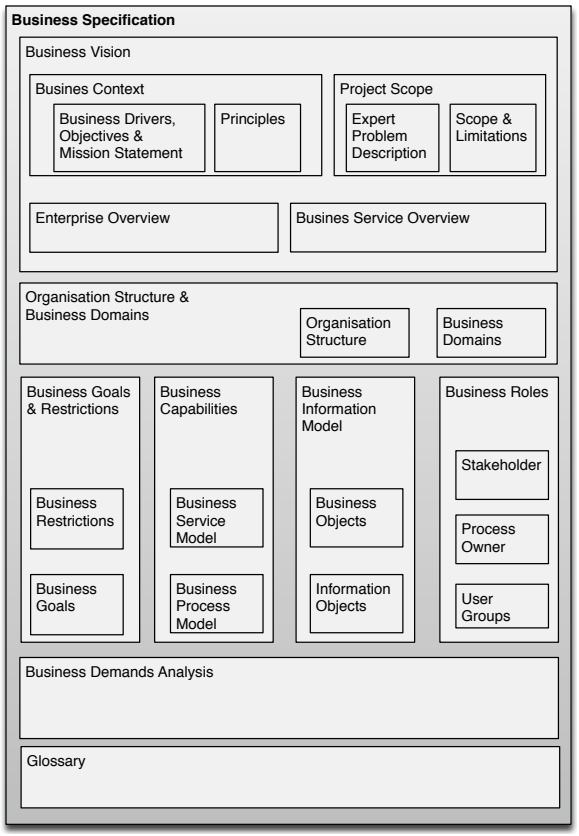


Figure 3.8.: Overview of Content Items of the Business Specification

3.4.1. Business Vision

The business vision introduces into the actual problems and defines the resulting scope of the business that shall underlie a change as part of a project. The business vision steers customer-side decision-makers 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 finally resulting (re-) structures of envisioned organisations.

Figure 3.9 illustrates the content items of the business vision (on the upper part) and included concepts. The bottom part of the figure sketches subsequent concepts that use the business vision as an input and that are initially summarised by the system vision as it is the case with the business services.

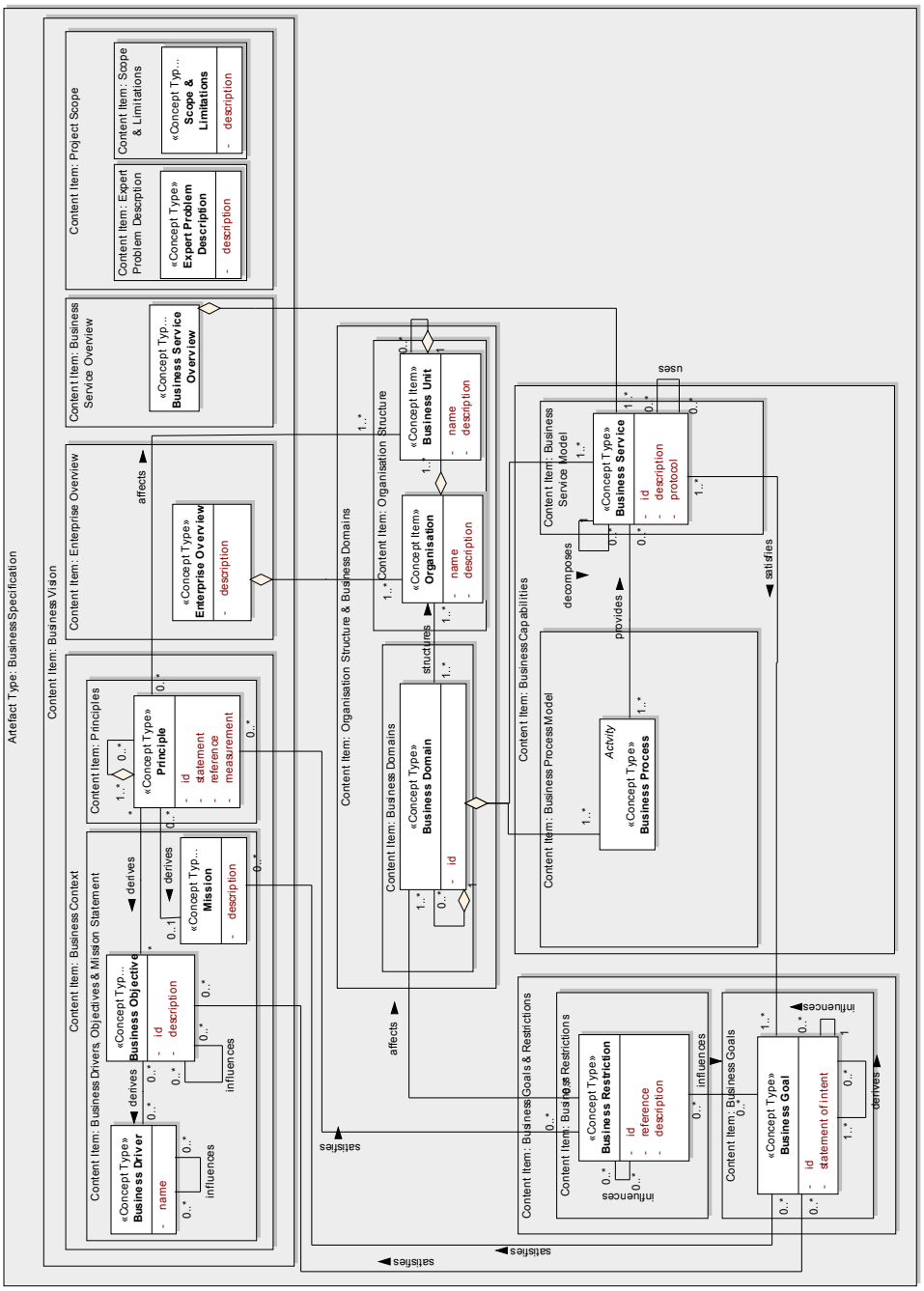


Figure 3.9.: Overview on *Business Vision* and underlying Concepts

3.4.1.1. Business Context

The business context describes the context of the organisation. For instance, the industrial branch and the incised market, and resulting cornerstones of a corporate philosophy. It summarises:

1. steering (mostly market-driven) drivers, objectives and mission statements that serve as a basis for defining the business goals.

2. the principles of a company serve as a basis for the business restrictions (e.g. policies like security guidelines).

3.4.1.1.1 Business Drivers, Objectives and Mission Statement

The business drivers, objectives and the mission statement are statements of direction that characterise the organisation and its underlying business. We distinguish between three concepts within one particular content item.

Organisations are subject to internal and external factors. These factors take effect on what business services are offered e.g. to a market and consequently take effect on the activities that are performed in order to provide the services. The impact of the environmental factors towards the organisation are the *business drivers*. They motivate ("drive") any kind of activity within a business, including customer / market needs and competition [BAB09, Gro]. An example is to outsource a specific number of business units and distributing them over several countries while delimiting to exactly one supplier, because the market "demands the products from one hand". Business drivers, however, can influence other business drivers positively or negatively.

Definition: *Business Drivers*

Business drivers are internal and external factors, including individuals, knowledge and conditions that initiate and support the design of business activities.

Business objectives are planned initiatives that concern major achievements of an organisation, typically within specific timeframes. For instance, increasing the customer satisfaction by reducing customer complaints. Business objectives are based on taking advantage of specific opportunities that are defined within strategy papers, business cases and the like.

Definition: *Business Objectives*

Business objectives are statements of intent that define planned outcomes to an organisation.

However, the business objectives are finally aligned with one *business mission* that steers the whole business, such as by stating an aimed market lead for specific products [Gro].

Definition: *Business Mission*

The *business mission* is a statement of direction and goal for the overall business.

Content Dependencies. The main input for defining the business drivers, the business objectives and finally the underlying business mission can vary from detailed strategy analyses and business cases, over existing policies to informally written bidding documents.

Furthermore, business drivers can be derived from business objectives, while these in turn can be derived from the principles.

The concepts that make use of upperly described ones, are the business goals (see content item 3.4.3.2). Business goals can be derived from the objectives and define "what is internally needed to achieve the objectives" such as increasing the

process lead-time within the business processes, thus, business goals satisfy the stated overall objectives. As the business mission is the main steering force of the business, the business goals consequently satisfy also the mission.

Syntax. The business mission is textually described as an informal statement. The business objectives as well as the business drivers can be described as:

1. graphs (often referred to as e.g. "drivers map") that benefits the illustration of the dependencies between the single statements
2. textual statements within a list.

How to specify the concepts strongly depends on the complexity, i.e. the amount of the statements and the amount of dependencies. If stating the objectives and the drivers within a table, it should be organised by first defining the mission(s), listing then the drivers and finally listing the main objectives.

3.4.1.1.2 Principles

The principles result from the business context and describe restricting directives of the current or the future state of a business, cross cutting several business units or the business as a whole.

According to [Gro], we define the principles as follows:

Definition: ***Principles***

A *principle* is a statement of belief, approach or intent which serves as a basis for directing the formulation of architectures.

An example for a principle would be the restriction of all the business units that act in Germany to the security standards "BSI Grundschutz manual", due to the business context and the targeted market of the company.

Each of the defined principles captures besides the statement itself a reference to external sources (e.g. to the manual), the information on affected business units and finally a measurement. The latter defines a criteria which serves as a reference for deciding on the appliance of business processes and information systems to the principles. Finally, principles can be decomposed into further principles, some of which may affect selected business units.

Content Dependencies. Principles can be derived from business objectives. Additional sources are business strategies, existing policies and technical specifications of current information systems, respectively during workshops or discussions about the like.

Furthermore, each of the principle affects at least one business unit of a company (see also content item 3.4.2.1). Finally, the principles can serve as a basis for defining concrete business restrictions, such as business rules and constraints (see also content item 3.4.3.1).

Syntax. Principles are informal statements that can be hierarchically decomposed. However, they can be expressed in a tables but if facing decomposition, it has to be clear that these should also capture the parent-child relations.

3.4.1.2. Project Scope

The project scope:

1. defines the problems of the business and included information systems within an expert problem description
2. concludes with the main objectives of a project without emphasising any kind of solution that will solve the problem.

3.4.1.2.1 Expert Problem Description

The expert problem description defines the (potential) problems that arise from the current state of the business from the perspective of a domain-expert. It reasonably introduces customers into the actual problems and thereby lays the foundation for defining the scope and the limitations of a project. For instance, heterogeneous historically grown applications (or landscapes) that lead to inflexible business process structures, actual maintenance problems of systems, etc.

Although the expert problem description is a core part of every business specification, the concept remains to a simplified description of the problems, while this can not be expressed with standardised (modelling) techniques or predefined contents.

Content Dependencies. The content itself does not have any dependencies to surrounding ones. It is based on several informal sources, such as bidding documents, system specifications, or documented gap analyses regarding business processes and security aspects.

Syntax. The problem description should be according to the target groups textually phrased with short statements. It explains, as clearly as possible, what the domain-specific problems are and if reasonable what actions can be taken.

3.4.1.2.2 Scope and Limitations

The scope and limitations provides:

1. the scope of the project that is necessary to tackle the identified problems. For instance, a necessary re-design of the business processes, a displacement of legacy applications or outsourcing of specific business processes.
2. any kind of limitations that have to be clearly stated. For instance, the elaboration of business restrictions that are provided by third parties and thereby are out of scope.

In any case, it does delimit from any solution ideas as these are the outcome of the project themselves.

Content Dependencies. The concept itself does not have any dependencies to surrounding ones.

Syntax. The scope and limitations should be described textually with clear statements, for example within a list. If reasonable, a differentiation between statements concerning the scope and ones concerning limitations can be made.

3.4.1.3. Enterprise Overview

The enterprise overview summarises the organisations and the included business units that are affected by a project. According to[BAB09], we define the concept as follows:

Definition: ***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.

Note that there is a difference between an *enterprise* and an *organisation*. An enterprise may encompass more than one organisations as more than one can be affected by a project. For example, within a development project that concerns the coupling of two information systems that cross-cut several business units of two different organisations, we address one specific enterprise (architecture). Each of the organisations are still autonomously defined, described with proper boundaries and under management of single individuals and / or institutions.

However, the concepts of the single organisations and their business units are defined in detail in content item 3.4.2.1. The focus lies therefore on the informal description of what organisations and what business units are affected, rather then on defining a holistic structure (i.e. an enterprise architecture) and corresponding responsibilities regarding participating roles.

Content Dependencies. The relations are given by the organisations that are included (at least one).

Syntax. The syntax strongly depends on the amount of organisations that are affected by the project. Within complex projects, the overview includes a brief textual description of the involved organisations (including their role within the project), while capturing it optionally within a table.

3.4.1.4. Business Service Overview

The business service overview summarises the business services of a customer that are or will be affected by a project. It includes existing business services being displaced or modified or new business services that will be implemented.

Each business service that is summarised is described in detail within the business service model in the content item 3.4.4.1.

Content Dependencies. The relations are given by the business services that are included (at least one).

Syntax. The representation of the business services within the overview can be constructed in two ways (or a combination of them):

1. as part of a detailed model (or graph) emphasising the dependencies in-between the single business services
2. within a textual description in natural language, e.g. in a list or a table

3.4.2. Organisation Structure and Business Domains

The organisation structure gives an overview on the hierarchical structure of the envisioned organisations. The organisation is clustered into business units that define major areas of responsibilities to achieve some outcome of value. In addition, business domains structure the business into major functional areas that are in scope of the architecture. Both can strongly differ from each other, as business domains cluster all the business processes (of perhaps several business units) in relation to exactly one process owner.

Figure 3.10 illustrates on the upper part the corresponding content items and encompassed concepts. The bottom part of the figure illustrates subsequent concepts that exhibit interdependencies with the concepts of either business units or business domains.

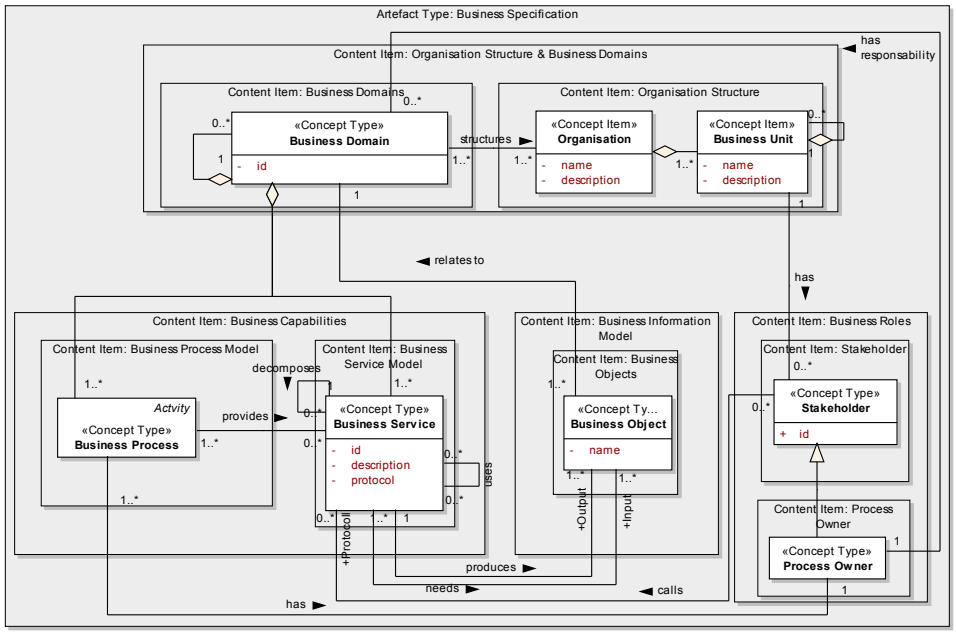


Figure 3.10.: Overview on *Organisation Structure and Business Domains* and underlying Concepts

3.4.2.1. Organisation Structure

The organisation structure defines the current state of single organisations and included business units. It benefits the determination of key reporting lines and in conjunction with a cultural analysis will indicate where key stakeholders are positioned [Gro].

We define the concepts as follows:

Definition: *Organisation* and *Business Unit*

An *organisation* is an autonomous set of hierarchically structured business units with clearly defined boundaries.

A *business unit* is a recognised part of an organisation under management of at least one stakeholder.

The organisations, as well as the included business units capture unique identifiers and a description of their purpose, respectively their contribution within the business. Finally, each business units includes the description of corresponding stakeholders. We refer in particular to the ones that are of major interest within single business units in terms of directly or indirectly participating within the project.

Content Dependencies. The only relations to surrounding concepts is given by single organisations that build part an enterprise (see content item 3.4.1.3) and from business units to participating stakeholders (see content item 3.4.6).

Syntax. The structure of the organisation can be defined using an organisational diagram.

3.4.2.2. Business Domains

Business domains define independent of the organisation's structure major areas of logically related business processes, i.e. a specific segment of a supply chain. Business domains are furthermore related to one process owner (an individual that is responsible for the domain). We use the term *business domain*, as *domain* is used in further contexts such as when referring to a domain of application. According to [EHH+08], we define business domains as follows:

Definition: ***Business Domain***

A *business domain* groups a set of logically related business processes and their sub-processes, thereby the business services that are provided by the business processes. It defines a major area of the business that is in scope of the architecture. A business domain can be decomposed, but relates in any case to exactly one process owner and its business processes.

Business domains can vary in their range, covering a large amount of business processes or a small one, as long as they relate to one process owner. Furthermore, they formulate structure from a functional perspective that can cross-cut several business units, as they cluster business processes that underlie high cohesion.

Hence, the business domain include the description of the responsible process owner, the encompassed business processes and further information system requirements that result from the support of the business processes.

Hint: *"Problem Frames", "Viewpoints" and the relation to Business Domains.*
There are two related approaches: *Problem frames* [Jac01] and *viewpoints* [KS96]. Problem frames offer formal means of structuring the environment that surrounds the future application into so-called frames with a set of pre-defined patterns. Instead, viewpoints are different means of communication that structure requirements in relation to single stakeholders and their attitude towards the application. The least common denominator are the business processes that have to be supported by the application (structured into business domains). They represent the main interest of single stakeholders (or are directly affected by such) and at the same time they are can be seen as a specific subset of problems to be solved. Thus, the business domains can be compared to problem frames that structure the business processes of a customers and at the same time as a view point as the business domains are related to a specific process owner.

Finally, business domains can be taken by following design activities as an orientation for structuring the logical architecture into components and thereby defining an architecture that functionally fits into the business process logic. Obviously, this also depends on how the business domains are specified (see also the relations).

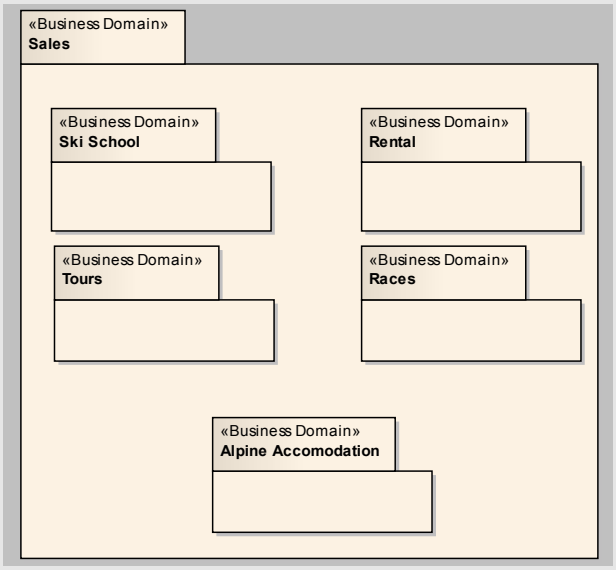
Note that as the business domains structure the content of the business specification and the requirements specification, they benefit also a structured process for business information systems' analysis (see also Sect. 3.9.2).

Content Dependencies. Business domains exhibit relations to one process owner that takes the responsibility for the domain and to the business processes and business services that they over-span. A further relation is given to the business objects to which the business domains relate as a consequence of grouping business behaviour. However, the formulation of business domains can be performed according to the business processes, the business objects and finally to the participating

business roles[2]. These information sets can be taken as an indicator for defining to what extend the business domains are built.

Syntax. The representation of the business domains can vary from an informal description of the single business domains within tables to the tool-supported representation using means like package structures (as exemplarily depicted below). In the latter case the packages themselves represent the business domains.

Example 3.1. *Representation of Business Domains*



3.4.3. Business Goals and Restrictions

This content item includes steering goals that have to be achieved by the execution of the business activities and further rules and constraints that have to be preserved during that execution. The latter describes constraints and rules that result from the principles of the organisation.

Figure 3.11 illustrates on its left part the corresponding content items and used concepts. The upper and right part of the figure illustrates excerpts of surrounding concepts that exhibit interdependencies with envisioned ones.

3.4.3.1. Business Restrictions

Business restrictions define any kind of restrictions towards the business and the underlying IT infrastructure. The restrictions arise from the principles (content item 3.4.1.1.2) and any kind of external influences that affect an organisation, such as laws. Business restrictions define how, respectively to what extent these influences take effect. We distinguish in particular between two kind of restrictions:

1. *Business constraints* that define not negotiable limitations for any kind of solution regarding business processes and information systems, i.e. aspects of

2 Similarly, logical components can be defined at the basis of same information, building them respecting the functions that they encompass, information that they process and finally roles / actors to which they interact.

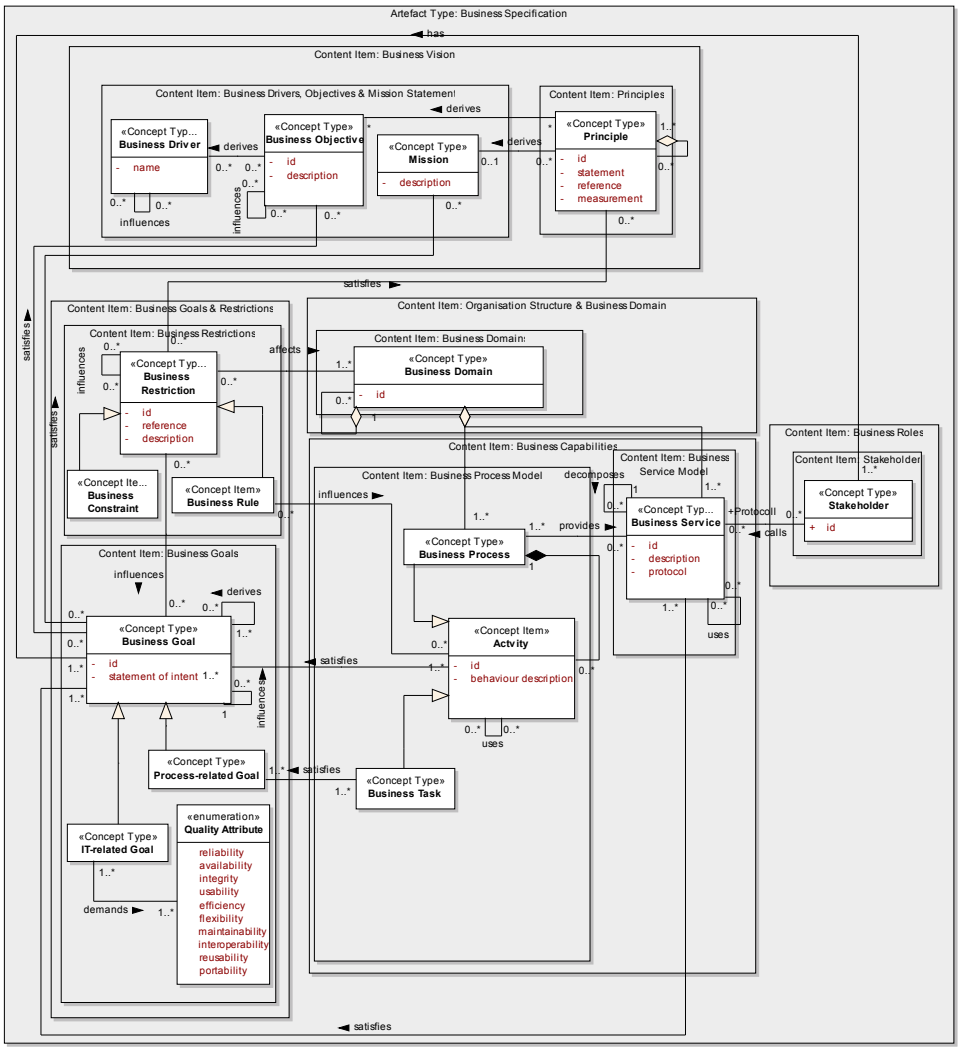


Figure 3.11.: Overview on *Business Goals and Restrictions* and underlying Concepts

current states that cannot be changed by the development of an information system.

2. *Business rules* that exclusively govern business process logic, i.e. activities in terms of defining them, constraining them and finally enabling them.

The business rules arise from are internal influencing factors and principles such as standards procedures. The rules take effect on the activities that are performed within a business. According to [Gro00, BAB09], we define the business rules therefore as follows:

Definition: *Business Rule*

A *business rule* is an actionable directive that defines or constrains 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.

In general, business rules exhibit the nature of action enablers triggered by a certain condition (comparable to "if then else" - statements) [Wie03].

Compared to business rules, business constraints define any kind of restrictions on the business process logic and further technical restrictions. They define operationalised external influencing factors, such as laws, IT security guidelines or technical design notes. They are not negotiable and are often stated independent of any business goals and their effect towards quality.

Definition: *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.

Both business rules and business constraints include following information:

- ID: A unique identifier.
- Reference: The reference indicates to the origin of the business rule, such as a specific IT security guideline, a law or a principle (see content item 3.4.1.1.2).
- Description: The description summarises the demands that result from the business restriction.
- Area of Validity: The area of validity defines the affected business domain (or the business as a whole).

Content Dependencies. Business restrictions can be derived from principles. Within complex organisations the restrictions are available within specific catalogues, such as within given security guidelines. A further relation is given towards the business domains in which the restrictions take the effect. For example, an internationally distributed organisation may define business restrictions that only take effect on business processes within a specific country and that have a specific concern ("For all business processes within 'Manufactoring' that act in Germany, the restrictions of 'IT Security Guideline [xy]' must be obeyed").

Finally, the business rules can affect in particular only activities that are performed within an organisation.

Syntax. Business restrictions are mostly defined within tables. The description of the single restrictions, i.e. the restrictions' assertions, remains to a textual representation. In the case of business rules, they can be alternatively to a representation in natural language expressed by means of predicate logic, in some cases extending to expressions using temporal logic due to their actionable nature. Although a formalisation benefits a unambiguous interpretation, it should be clear that the target groups (the customer-side stakeholders) must understand and accept the rules.

Subsequent example illustrates the representation of business restrictions within a table.

Example 3.2. *Business Restrictions*

ID	Type	References	Description	Area of Validity
BC1	Business Constraint	IT Security Guideline	Personal data of employee must not contain the religious affiliation.	Customer Management, Human ressource management
BC2	Business Constraint	SPO Sales	Each Customer has a unique identifier.	Sales, Customer Management
BC 3	Business Constraint	IT Dev. Guideline	System development projects must be realised only including suppliers that can guarantee at least five years of support.	-
BC 4	...	...	...	...
BR 1	Business Rule	SPO Sales	If a customer books courses, he must be directly informed about the time schedule and the meeting place number.	Sales
...	...	...	...	...

3.4.3.2. Business Goals

The business goals need a special consideration as they state specific intents of the business at some future point in time. They motivate the activities of the business and finally result in an expected business value [RR99]. When stakeholders perform their business processes and state their requirements, they have certain business goals in mind that have to be justified. A business process or a requirement without a goal is perhaps a statememt without any value to the customer.

The following definition of the concept is made according to [vL03, vL04, vL09, EHH+08]:

Definition: ***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:

- Process-related goals: The satisfaction of these goals lies on all the activities that are performed within an organisation (business processes and business tasks).
- IT-related goals: they directly demand high-level properties and (quality) characteristics of an information system in order to support the process-related goals.

Business goals, however, may contribute to a certain business objective and / or a mission of an organisation (see also content item 3.4.1.1.1). They must at least be related to one stakeholder that benefits from the business goal. We distinguish according to [EHH+08] during decomposition between three different abstractions depending on the goals' assertions: business goals in general and process-related and IT-related ones in particular.

Business goals have mainly financial character and affect the whole organisation, such as demanding a cost-reduction within the business.

When decomposing them to more fine-grained goals they affect not only the overall project but in particular the business processes. For instance, by demanding a

decrease of a process lead time in order to save costs during execution of the business processes. These *process-related goals* are used to steer the business processes into the desired direction. They help in translating the actual business process landscape into a necessary one that enables a customer to reach his business goals.

Finally, *IT-related goals* can be derived from such process-related ones and directly address high-level quality characteristics of the information system. For example a demanded increase of the efficiency or the maintainability. IT-related goals relate to one or more quality attributes as stated by the ISO Std. 9126 [ISO03] and the ISO Std. 25030 regarding (product) quality requirements [Boe08]).

Definition: ***Quality Attribute***

A *Quality Attribute* reflects a composition of perceptible characteristics that are exhibited by an application and its environment due to its properties. Quality attributes are hierarchically organised. An example is “Efficiency” that can be decomposed into “time behaviour” and “resource utilisation”.

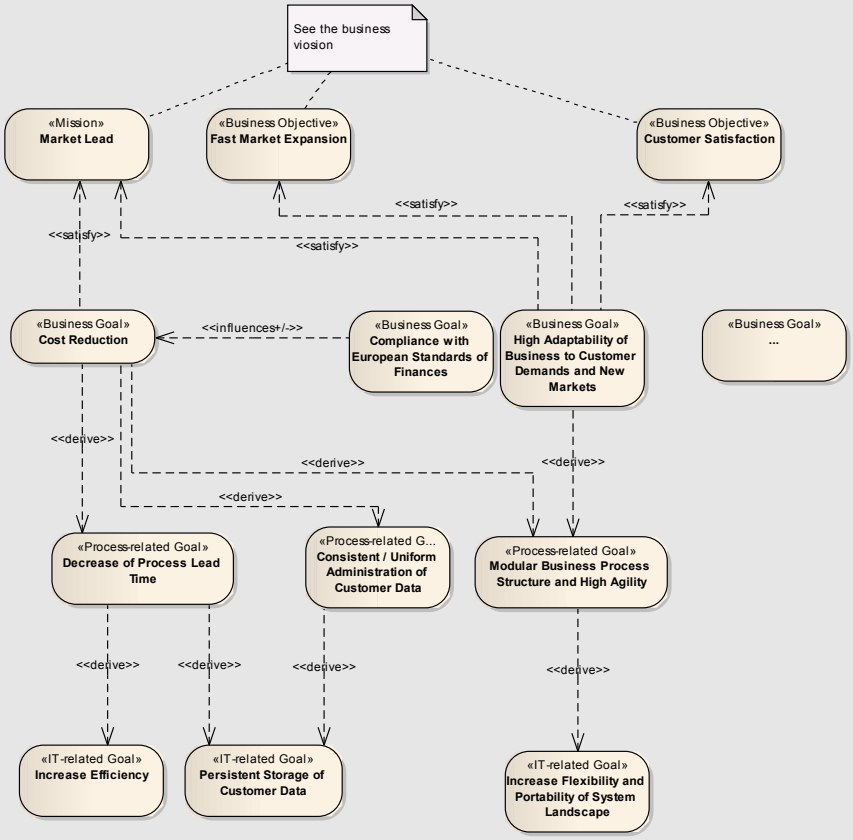
Hence, IT-related goals are in the primary scope of especially system quality requirements that often can be directly derived from such goals (see also the content item 3.5.3.5).

Content Dependencies. Business goals can be derived from business objectives and from the mission (see also content item 3.4.1.1.1), what in turn is to be captured within the goals. Furthermore, the business goals can affect further concepts like business processes. However, this is to be defined within the affected concepts as a rationale and not within the goal.

In addition, business goals may influence other goals by supporting them other or by being in conflict with them.

Syntax. Due to the nature of business goals that can be derived from other goals and due to the resulting influences, business goals can be described with goal graphs. A prominent approach that makes use of graphs is KAOS (Knowledge Acquisition in Automated Specification of Software) [vL03, vL04, vL09]. A second alternative is given by defining the goals within tables in which the influence and derive relation is captured in addition to the statements themselves. Although, graphs offer the advantage of illustrating the influences, this only makes sense in complex projects that justify the management of the graphs. Subsequent example illustrates the depiction of business goals within a graph according to the concept model .

Example 3.3. *Business Goals*



3.4.4. Business Capabilities

The business capabilities include:

1. the *business service model* describing logical representations of repeatable activities that have a specific outcome (like the service "stock ordering") [TOG09].
2. the *business process model* emphasising the activities themselves that have to be performed in order to provide the business service, respectively when referring in general to the service (like the executed activities within the business process "order stock").

Figure 3.12 illustrates the corresponding content items and the concepts. The centre of the figure illustrates the concepts of the business process model and the business service model. The surrounding parts of the figure illustrate excerpts of surrounding concepts to which we refer when constructing the envisioned ones.

3.4.4.1. Business Service Model

The business service model describes single capabilities (and their relations) of a business by means of a logical representation [TOG09], i.e. a blackbox description of the work that is carried out as a result of a specific set of activities[BAB09].

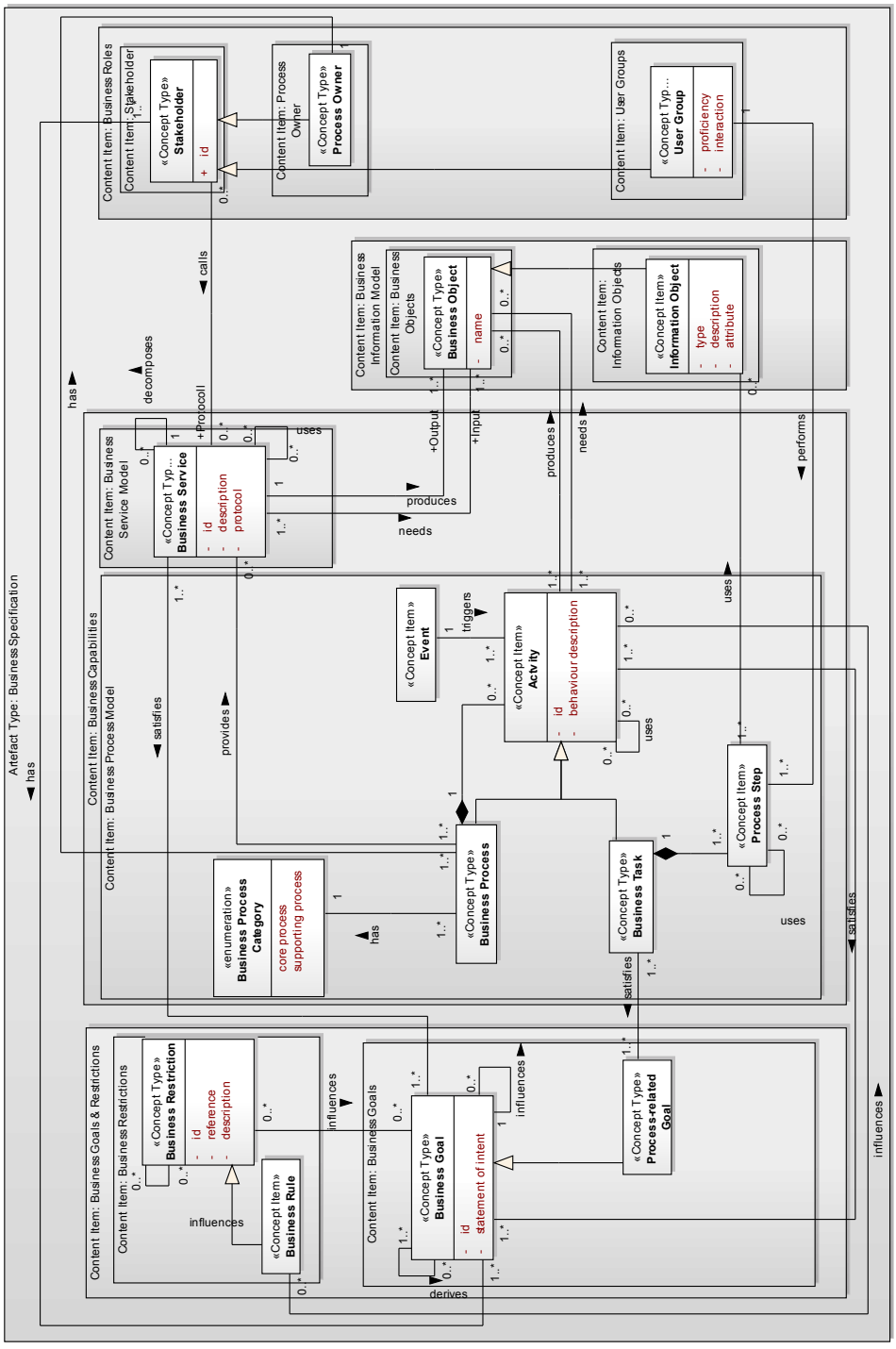


Figure 3.12.: Overview on *Business Capabilities* and underlying Concepts

Note that we distinguish between business services (“what is done within the business?”) and information system services (“how is the work supported by the system?”, see content item 3.5.3.3.3), while none of both are directly related to specific technologies like web services.

We describe with the business service model what services an organisation internally and externally offers. For instance, the service "Travel expense accounting" or "Travel booking", while the first could be internally offered to business units and employees and the latter could be externally offered to a market. Business services benefit a logical representation of valuable capabilities without the necessity of directly describing the activities that are performed in order to provide the services. They serve in addition to the specification of business processes as a representation that emphasises structure rather than details of behaviour. This is for example important to projects concerning re-design and outsourcing parts of an organisation since the realisation of the "internal workflow" of single services is then not in scope.

However, each business services logically represents a set of activities that modify a set of business objects (see content item 3.4.5). According to [BKM07], we define the concept as follows:

Definition: ***Business Service***

A *business service* defines a business capability provided by the execution of one or more business processes satisfying at least one business goal. It can be hierarchically decomposed and maps an input (stimulus) to an output (response).

Furthermore, business services are offered by a service provider within an organisation and can be called by a service requester (e.g. a target group within a market), both conceptually represented as stakeholders (see content item 3.4.6.1). Each business service includes besides an identifier and a (service) description a collaboration contract that optionally defines the sequence of steps in which a service can be used by the requester. This contract is conceptually expressed by the attribute protocol.

Hint: *Collaboration Contracts and Quality of Service*
Note that it is possible to define within the construction of business services explicit collaboration contracts that describe besides other characteristics of the business services specific sequences of usage and the quality of service, e.g. demanded availabilities. We especially make use of these means when describing information system services that define how the business services are at least partially automatised, i.e. supported, by the IT. However, within the definition of business services these information can be captured within the "Protocol" of the services, depending on the project complexity and the scope. If it is necessary to define concrete collaboration contracts, the concepts of the contracts (including the dependencies) of the information system services can be transferred to the business services (see also content item 3.5.3.3.3).

Finally, each of the business services include the following information:

- ID
- Description of their purpose and content
- Business goals that they satisfy
- Other business services that they use and / or that use the envisioned one.
- Protocols that (optionally) describe expected sequences of inputs and outputs.
- Business objects that are processed in terms of a needed input and a generated output

Content Dependencies. There exist several relations between business services and surrounding concepts. Although the concept of a business service (e.g.

"Course Administration") and the one of a business process, respectively activity (e.g. "Administrate Courses") exhibit similarities, they are not the same. Instead, they relate to each other. Both:

1. are refined and may collaborate with other services, respectively processes (they may "use" each other implicating workflow).
2. define for a specific input an output, thus they describe the behaviour by describing how they operate on a set of business objects and information objects (see content item 3.4.5).
3. must satisfy at least one business goal.

Still, a business process is more an "internal" representation of the business service in terms of a sequence of steps that are performed in order to *provide* the business service. Note that each step within a business process model can be also seen as an (atomic) service that cannot underlie further decomposition. Still, within the description of the services and processes, we make use of different concepts. Regarding these concepts, a business service does not include a mandatory description of the stakeholder that can call the service. The service is autonomously described without necessarily involving any kind of interactions and thereby abstracts from the business processes.

Finally, the dependencies to information system services that "realise" the business services (by defining how systems' functionality support the business services) are described in content item 3.5.3.3.3.

Syntax. According to the established use of event/response tables [Wie03], respectively I/O-tables[HT09] for representing the input/output-relation, these tables can be extended by upper information in which the columns describe the information and the lines each service. Each business service can (and should) be structured by the business domains in which the service is provided (see also content item 3.4.2.2).

3.4.4.2. Business Process Model

The business process model describes the (idealised) activities that are performed within an organisation in order to provide the business services. The business processes serve as a basis for understanding what business behaviour underlies single services and how this behaviour can be supported by future information systems (what is initially depicted within the system vision of the requirements specification in content item 3.5.2).

According to Thurner [Thu04], a business process is characterised by:

1. Encompassed activities: A business process consists of a set of activities that are executed and that define the behaviour.
2. Sequence of executions: 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.
3. Processed information: During the execution of the activities, a specific set of information is processed in terms of being created, read or modified (see also the business information model in content item 3.4.5).
4. Parcitipating roles: Business processes are defined and administrated by specific roles and also performed by specific roles (see also the business roles in content item 3.4.6).
5. Motivating business goals: Each activity has to satisfy at least one business goal (see also the business goals in content item 3.4.3.2).

As already described as part of the artefact abstraction model in section 3.2.3.1, business processes are decomposed over two different levels of abstraction, including (1) a structural view onto a taxonomy within the business process hierarchy and (2) a view onto the internal workflow description within the business process logic. We conceptually distinguish therefore between:

- business processes in general that are defined and administrated by process owners in order to achieve a business goal and for each business process
- a set of encompassed business tasks providing each a description of atomic process steps that are performed by user groups in order to achieve process-related goals .

Note that compared to the approach of Cockburn in [Coc00], a business task corresponds to a "black box business use cases" while the description of the encompassed process steps corresponds to a "white box business use case".

We define in the context of this report the concepts of business processes according to [Thu04, Coc00, EHH+08] as follows:

Definition: ***Business Processes and Business Tasks***

A *business process* is a cross-functional sequence of activities across multiple positions in order to achieve a planned work result in an organisation and that contributes to at least one business goal. It is administrated by a process owner and serves to directly or indirectly produce a product or provide a business service for a customer or a market. The activities are triggered by an event[3] and may use other activities to transform (produce, read and 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 sequence of atomic process steps, while these in turn are:

- performed by exactly one user group aimed at achieving at least one process-related goal and
- can be supported by an information system.

While business processes exhibit the same granularity as business services that they provide, the included business tasks define their *internal realisation*. An example for a business process could be "Administrate Courses" that provides when executed the business service "Course Administration". This business process might be decomposed to the business tasks "Add Guest to Course", "Select Trainer for Course", etc. while these business tasks would be performed by the user group "Administrator". Each of the business tasks, in turn, includes a sequence of atomic process steps (like "Search Trainer") while one process step is performed by exactly one user group, including a reference to selected business objects that are read or modified. For instance, the business object "Guest", including within the corresponding content item 3.4.5 its information, such as "Club-ID", "Full Name", "Age", etc.

Note that from the business task descriptions, the use cases can be derived that then describe sequences of interaction between future users and the information system. While the business tasks provide a description of the workflow that is generally performed, a use case describes how parts of this workflow (and consequently the business service) can be supported by a system. Hence, the process steps within the business tasks are the ones that can be realised by user actions

3 A change of state or a temporal event.

or system actions (e.g. the mentioned one "Search Trainer" that can be automatised by a system that administrates available trainers and related courses). Still, what business tasks are finally supported and how this support shall be realised is described within the requirements specification, in particular within the system vision in content item 3.5.2 and in the use case model in content item 3.5.3.3.1.

Therefore, the definition of the three levels of abstraction within the business process model including three different concepts benefits a clear stop criterion for decomposition of single business processes until reaching the level in which one user group performs single atomic steps. This enables a clear transition to the use case model and thereby the analysis of what single steps shall be supported in what way by the system.

Finally, as depicted by the concept model in figure 3.12 we distinguish furthermore during business information systems' analysis between two different categories of business processes: *core processes* and *supporting processes* (see also [EHH+08, WGWP02]).

Definition: *Business Process Category*

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*.

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. For instance, in the example of the business process "Administrate Courses", this business process is a core process as it is the envisioned one that is meant to achieve some outcome and that shall be at least partially supported by an information system. A supporting process could be one from the area of financial management (like accounting). It is not directly involved and perhaps not in scope of this particular information system, but still necessary[4]

Supporting processes are in general not in primary scope of development projects and are usually not modified as a consequence of the like. Still, they have to be taken into account respecting their activities and their business objects.

Content Dependencies. Business processes have dependencies with the business roles (content item 3.4.6), in particular with the process owners that administrate the processes and the user groups that perform the business tasks. The latter is captured within the business process model as part of a "pool" or a "swimlane" (depending on the chosen syntax), both being means that group a set of activities performed by the same role.

Another mandatory relation is given by the business goals that are satisfied by the business processes and the process-related goals that have to be satisfied by the business tasks. Although the goals are described in detail in content item 3.4.3.2, the process descriptions should provide which of the goals they satisfy. Similarly, the business processes should define which business objects and information objects are involved and how they are involved (being read, modified or created).

However, a further relation can be given by a business rule that is described in content item 3.4.3.1. They can take effect on the definition of the business pro-

4 When developing for example information systems for financial units, the categorisation could be performed vice versa, i.e. the financial processes being in scope of the project and thereby a core process to be modified and supported.

cesses, respectively the activities. In that case, the relation must be also captured as a rationale within the business process.

Finally, the last relation is given to the use cases (and is captured within the like). Use cases describe how single business tasks can be supported, thereby they make use of specific business tasks (see also content item 3.5.3.3.1).

Syntax. When choosing the syntax for representing the business processes we can in general distinguish between notions that emphasise data flows, i.e. the processed business objects, and notions that emphasise control flows within and between the business tasks of a process. We recommend to choose for a notion that follows an orientation on control flows, as the business objects are also referenced in these diagrams, even if indirectly. Cases that argue for data flow orientation are:

- if the organisation is already structured according to the business objects instead of according to functions (respecting the business units and / or the business domains)
- the business information system itself emphasises the data the has to be processed (the information system objects) instead of the work flow as it would be the case when developing workflow management systems.

We subsequently summarise in table 3.1 selected approaches respecting their characteristics, while a detailed description can be taken from [Thu04].

The following examples illustrates the representation of business processes with BPMN, showing first an excerpt of business process model and its relation to the business domains and the process owners, and afterwards an excerpt of a business task defining workflow.

Example 3.4. *Business Processes*

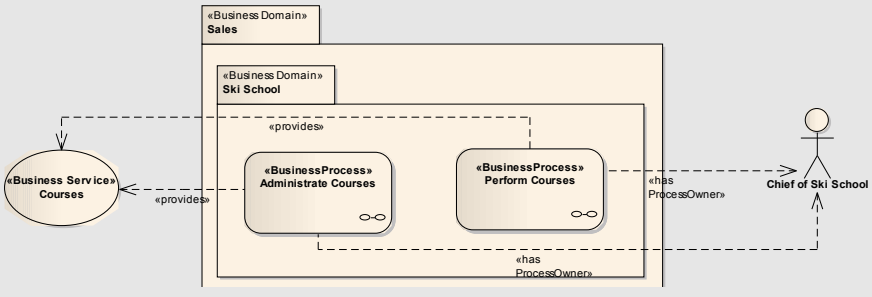
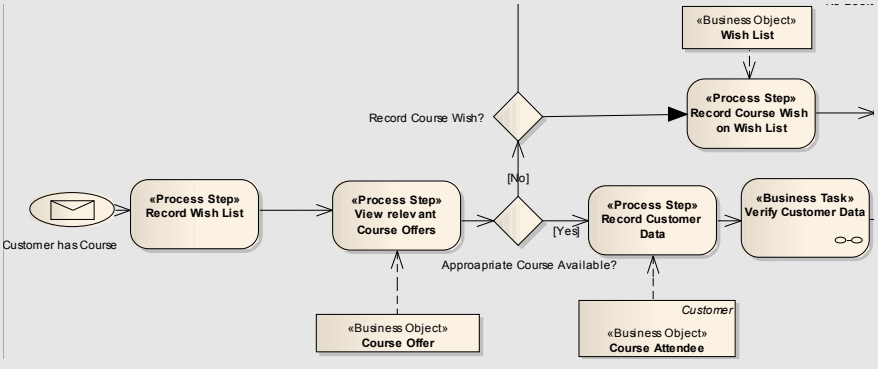


Table 3.1.: Syntax for the Business Process Models

Approach	Focus	Characteristics
Business Process Nets [Thu04]	Data Flow	Mathematical underlying semantic; good applicability as concepts can be completely expressed and enables clear transition to components of the system's architecture.
Data Flow Diagrams (Structured Analysis) [DeM79]	Data Flow	Low applicability due to missing possibility of applying business roles and events.
Message Sequence Charts (MSC) [(IT99]	Data Flow	Underlying process algebra, focus on (event-triggered) interactions between business roles, thereby good applicability.
UML Sequence Diagrams [OMG07a, OMG07b]	Data Flow	Informally described semantic, but including consistency constraints and possibility of extending syntax. Good applicability as diagrams are readable, demanding (like MSCs) interactions between business roles.
Event-driven process chain / EPC (ARIS) [Sch91, Sch02, SJ96]	Control Flow	Operational semantic with petri nets (in parts). Emphasise events and control flows in relation to business objects and temporal dependencies. The latter is reduced to sequential process chains, thereby exhibits low applicability.
FUNSOFT nets [DG96]	Control Flow	Completely mapped onto petri nets, thus supporting automatisation respecting tool support. Still, business roles not mandatory related to concepts of business processes and can not be visualised within this diagram. Hence, exhibits an average applicability.
UML Activity Diagrams [OMG07a, OMG07b]	Control Flow and Data Flow	Concepts can be completely expressed, offering for the representation a combination of both data and control flow. Semantic of concepts is informally described and in parts very ambiguous, e.g. neglecting consistency constraints. Hence, average applicability.
Business Process Modeling Notation (BPMN) [(DT03]	Control Flow	Semantic informally (textually) described. Concepts can be completely expressed but resulting diagrams become quickly complex.

Example 3.5. *Business Tasks*



3.4.5. Business Information Model

The business information model describes all the information that is processed and referenced during the execution of the business processes, respectively services. Similar to the decomposition of business processes over two levels of abstraction, we also distinguish within the business information model between two content items:

- *Business objects* that are focused within the business processes hierarchy: "What goods and information do I need to set up a business?", e.g. an order within a restaurant.
- *Information objects* that represent the processed information itself within the business process logic: "What information is interchanged between user groups in order to perform their business tasks?", e.g. the information that the waiter has in his mind when taking the order.

Figure 3.13 illustrates on the right side the concepts of the business information model and their dependencies.

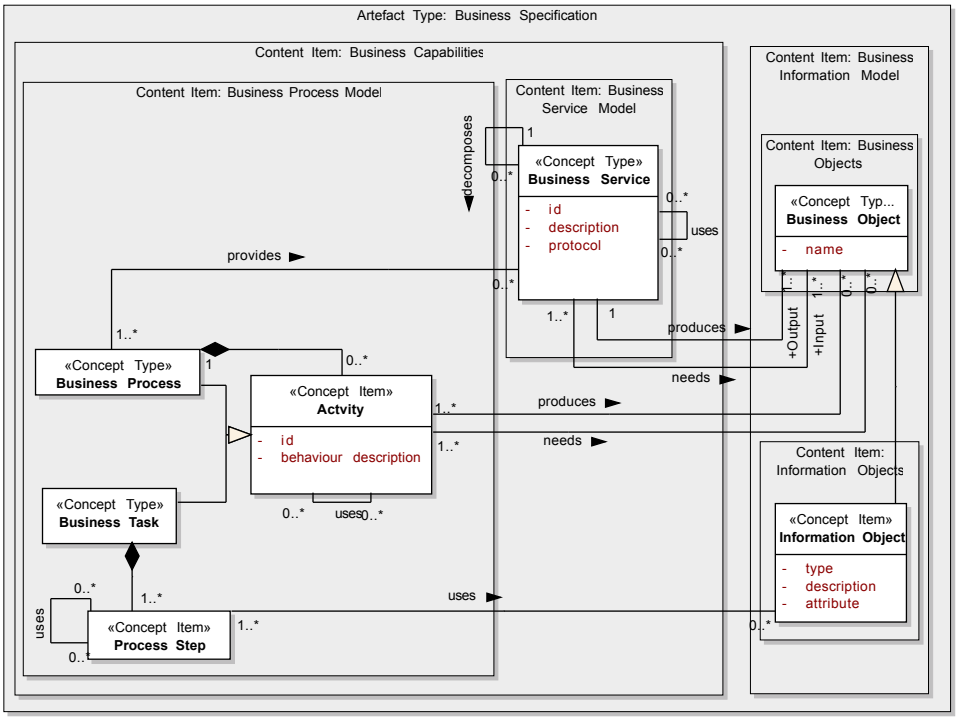


Figure 3.13.: Overview on *Business Information Model* and underlying Concepts

3.4.5.1. Business Objects

We conceptually define the business objects as follows:

Definition: *Business Objects*

A *business object* represents a unique object of reality. It can be of material (physical) nature, such as a "Car" or of immaterial nature, such as an "Order" that a waiter has in his mind.

Business objects are in most cases (real) goods that are produced and used as a result of a provided business service or as a result of an executed business process. Hence they are referenced when referring to an abstract description of the taxonomy. The information that is captured within the business object (that is of interest when defining workflow) is defined within the information objects.

Content Dependencies. Business objects are used by at least one business service and can be created and / or used by activities (business processes and business tasks). However, we do not have an explicit relation in-between several business objects themselves. The relations are implicitly (therefore optionally) given, because they arise as a consequence of the input/output-relations when being used by surrounding concepts.

Syntax. Within the business objects we do not necessarily define any relations. Hence, a diagrammatic representation of the business objects such as with UML class diagrams [OMG07a, OMG07b] emphasising structural dependencies is optional and depends on the complexity of e.g. the business process model (see content item 3.4.4.2). Same complexity takes effect on if to structure the business objects with business domains. However, as an alternative the business objects can also be summarised within a list.

3.4.5.2. Information Objects

Each of the business objects can be specialised by information objects that define the content of the business objects. Information objects are in scope of information systems as they represents the information object if the corresponding activity that refers to the object has to be supported by the system (see also the content item "system vision" in 3.5.2). We conceptually define the information object as follows:

Definition: ***Information Objects***

An *information object* describes the (information) content of a business object that is processed and that can be reproduced by an information system.

Compared to business objects, the information objects are of interest when defining a workflow description within the business process logic with business tasks. They capture the information that is needed as an input or produced as an output within an activity. For instance, the information that represents the order that the user group "waiter" has in his mind, like the "Table Number".
Each of the information objects captures:

- Type: A unique identifier that indicates to the type of the information object (to the corresponding business object such as "Order").
- Description: A short description gives a quick overview on the content of this information object.
- Attributes: The attributes describe and list the detailed content (items) of the object.
- In/Out-Relation (Optional): The relation to the activities that process the object.

Content Dependencies. The dependencies are the same as the ones of the business objects. In addition, information objects can be implicitly referenced ("used") by one or more process steps (within the business task description).

Syntax. Similar to the business objects, information objects have no explicit relations to each other, a representation of the like in terms of structural diagrams is not necessary (using for example UML class diagrams, entity relationship models, ...). Exceptions can be made depending on the complexity of the business process model and the project scope.

3.4.6. Business Roles

The business roles define all the roles that are directly or indirectly related to the business and that have a specific interest towards the project. We distinguish between

1. stakeholders including individual, groups and organisations
2. process owners, having the authority on business processes
3. user groups that represent individuals that directly operate in a set of business processes and might interact with business information systems

The roles serve for understanding the characteristics, responsibilities and finally needs regarding the future usage environment of an information system. Figure 3.14 illustrates the concepts for constructing the business roles on the right side, surrounded by further concepts that make use of the business roles.

3.4.6.1. Stakeholder

According to [Cle05, Smi, GBB+06] we conceptually define a stakeholder as follows:

Definition: ***Stakeholder***

A *Stakeholder* describes a unique individual, group or organisation that is related to a 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.

Stakeholders are customer-side or customer-related roles that have an interest in the project and that can mobilise resources to affect its outcome in some way [Wie03]. For example, the department of marketing as a customer-side role. Or a governmental authority as a customer-related role. Any kind of roles that have a certain attitude or influence on the project and the related information system. In the example of the governmental authority, this stakeholder can be involved into the project by defining specific laws that affect the project (see also the business restrictions in content item 3.4.3.1).

Content Dependencies. Each of the stakeholders that participate in a project has at least one business goal that should be satisfied. Furthermore, the stakeholders that are directly involved into an organisation can be allocated to exactly one business unit. However, as stated within the business service model in 3.4.4.1, stakeholders can be (optionally) involved into the execution of a business service. Finally, an informal (conceptually not expressed) source for deriving the stakeholders are stakeholder analyses documents.

Syntax. Stakeholders are textually represented in natural language, for example within a list. A way to structure the stakeholders according to e.g. environmental

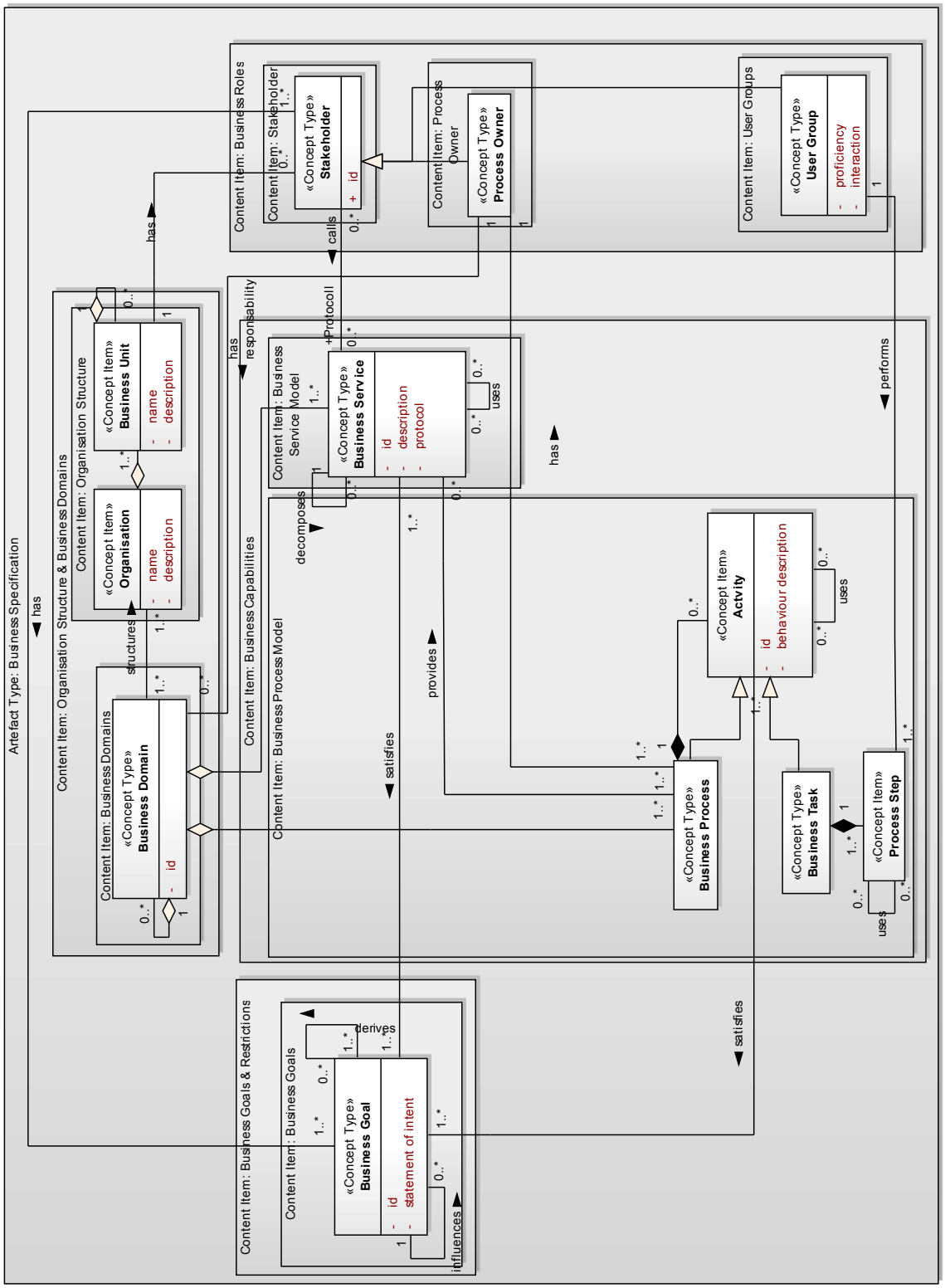


Figure 3.14.: Overview on *Business Roles* and underlying Concepts

aspects is to elaborate in addition to the list an overview and illustrating the stakeholders for example according to the onion model. Further ways of representing stakeholders are defined in [Rup07].

3.4.6.2. Process Owner

In addition to general stakeholders we define a specific subclass of them in terms of process owners.

Definition: ***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.

Content Dependencies. Compared to stakeholders, process owners have in addition a dependency to at least one business processes (and included business tasks) that they administrate.

Syntax. See the content item stakeholders in 3.4.6.1.

3.4.6.3. User Groups

While process owners define and administer specific business processes of an organisation (e.g. "Financial Management"), user groups are the individuals that perform the corresponding business tasks from an operative perspective (e.g. "Perform Annual Tax Calculation").

According to [KCH+90], we define the user groups as follows:

Definition: ***User Group***

A *user group* is a group of individuals with a specific proficiency directly performing a set of business tasks and interacting with information systems.

User groups, however, are an abstract collection of individuals with particular needs, interests, expectations, behaviours, and responsibilities that characterise the relationship between a class or kind of users and the future information system. They serve therefore as a primary source for deriving actors within the information system requirements within the requirements specification.

The main information is (1) the users' *proficiency* (e.g. "an expert in ..") and (2) theirs *interaction* (including the frequency, regularity, continuity, etc.). We focus on those aspects that enable a clear transition into the actors description within the requirements specification and furthermore enable a development of a system that is compliant to the expectations of real individuals. This supports also a perspective-based testing.

Content Dependencies. In addition to the dependencies of the stakeholder concept, one user group performs at least one business task, respectively single process steps. Informal sources for deriving user groups (i.e. not conceptually expressed) are workplace directives and existing authorisation concepts.

Syntax. See the content item stakeholders in 3.4.6.1.

3.4.7. Business Demands Analysis

The business demands analysis describes for each the the business services and necessary activities the:

- potential costs that are caused by implicated activities being performed by the individuals and the costs that arise from a system's development in order to support this activity.
- resulting business values that arises when achieving defined business goals.
- upcoming business risks that threaten the business.

These three indicators justify any customer-adequate decisions during BISA activities in terms of decomposition and modification of the current business processes and finally the decision on if to automatise envisioned business processes. The business demands analysis addresses the question "Can modifications of the process landscape and the automatisation of business process logic by information systems achieve a business value the stakeholders are ready to invest for?". Furthermore it serves as a basis for the prioritisation of resulting requirements (see also section 3.5 introducing the requirements attributes).

How profoundly to document the business demands analysis (or if having to extend it with further information) depends on project scope and further constraints. For instance, if it is demanded by customers as an economic feasibility study or if it is performed from an engineering perspective in order to justify any decisions that have been made during the elaboration of the business specification (and the requirements specification). We subsequently describe therefore only the principles in terms of the concepts that have to be created and surrounding concepts from which they depend.

Figure 3.15 illustrates the concepts and the dependencies to surrounding ones, also including the concept of the information system services (the system-side realisation of a business service) from the requirements specification.

Business Values

The business value of an activity is defined by its contribution to a company's success, e.g. the contribution to achieving a business goal like cost reduction. A very simplified measurement could be measuring the business value by ranking the contribution according to ranges like "low contribution" to "essentially contributes". However, the combination of the following five indicators can be taken as a possible measurement:

- Criticality: "What impact would it have, if the business processes could not be performed? Is it complex with many dependencies to other processes?" [Rup07]
- Access Frequency: "How often is the activity performed?"
- Effectivity: "How well does the activity support the customer in achieving the business goals (with and without system-side support)?"
- Compliance: "Is the activity enforced by a business restriction (from which the business goal depends on)?"
- Satisfaction: "How satisfied would the user groups be while performing their activities, e.g. if it is supported by the system?"

Costs

There basically exist two kinds of costs companies have to face: The costs for executing a set of activities by individuals and the costs that arise from the implementation and the maintenance of the information systems in order to support these

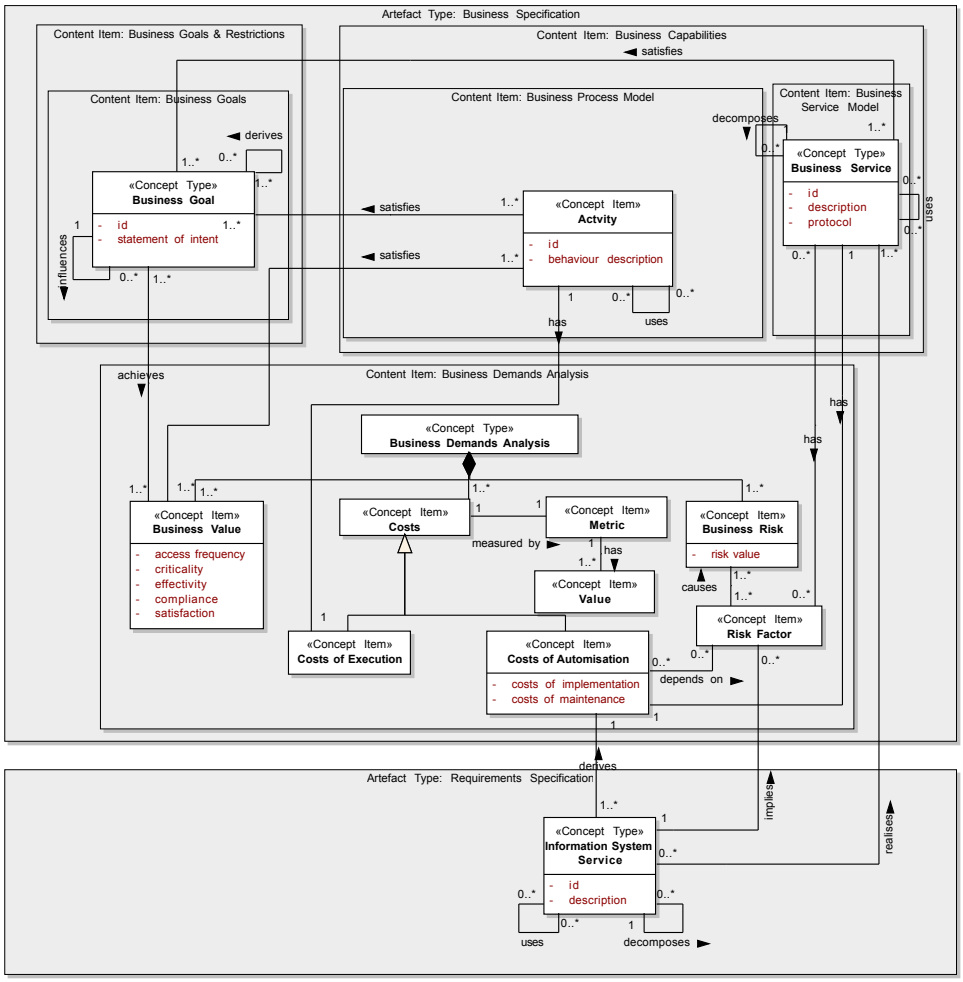


Figure 3.15.: Overview on *Business Demands Analysis* and underlying Concepts

activities. As not all these costs can be expressed by means of monetary values (especially regarding the execution of the activities) an appropriate metric has to be selected for the corresponding measurement (with specific values), referring in particular to activity-based costing. For instance, the number of process steps (with and without the support of a system), the process lead-time, etc. Regarding the costs of automatisation, especially the costs for realising a business service with an information system can only be approximated if having a certain knowledge on the concepts of the requirements specification. For instance, if having specified the use cases for the purpose of approximating the (functional) complexity referring to function points analyses as a basis for the costs estimation. However, note that these approximations still depend on many (technological) factors that often are not clear at the beginning of projects (see also the process model in Sect. 3.9.5).

Business Risks

Business risks can be calculated based on selected risk factors that cause potential risks to the business. We distinguish between factors that arise from the business services themselves (e.g. leading to the risk of a market rejection of a business

service) and risks that are caused by factors that arise from the system-side realisation of the business service (e.g. that not all features are delivered in time). Note that there exists no silver bullet for calculating the values for the risks that are implied by the factors. One possibility is, however, to define specific test criteria (in form of questions that address the different factors) including a rating of the risks according to their impact (or criticality e.g. in terms of resulting costs) and the probability that this particular factor occurs. Further information can be taken from [Isl09, IJH09].

Content Dependencies. The concept item of the business value has dependencies to the activities and the business goals. Each of the business goals achieve at least one business value while each activity is justified by the same business goal. This activity causes exactly one cost for executing it. If automatising this activity (respectively the corresponding business service) it causes one cost entry for implementing it and for maintaining it. Finally, each business risk is caused by at least one risk factor, while these can arise from the business services and / or their system-side realisation (the information system services).

Syntax. The business demands analysis should be defined within a table, while each entry illustrates the demands (a rating) for one business services. The representation of the values (regarding the business risks, the costs and the business values) depend on the chosen metrics and the project needs.

One example for a calculation of the entries could be rating the single concept items according to defined test criteria (questionnaires) and finally aggregating the values to an overall percentage for each of the business services. For example, by estimating the satisfaction within the business value according to "user would be frustrated" over "user does not care" to "user would be satisfied".

3.4.8. Glossary

The glossary defines project-specific terms that addresses a customer's domain and its underlying business process logic. As (1) terms that are familiar to contractors are often not familiar to customers (and vice-versa) and (2) the semantic of terms within one industrial sectors might differ from the semantic of same and similar terms within another industrial sectors, this concept defines an individual, but consistent set of terms upon which the following communication can be based on. This fundamentally enables:

1. an effective communication that is based on a well-defined vocabulary.
2. the prevention of unnecessary misunderstandings that might expand the time schedule and the budget of a project.

Note that the concept consists of one element with no dependencies to surrounding ones. However, we define the glossary as follows:

Definition: ***Glossary***

A *glossary* defines the semantic of specific terms used within a business domain and synonyms that can be used for the terms.

The business glossary consists of the following content:

- Term: Name of the term.
- Synonym in foreign language: A synonym that is semantically equivalent within a foreign language, e.g. German.

- Definition: Brief description of the semantic of the term.
- Synonyms: A list of terms that can be used with an equivalent meaning.
- Abbreviation: Abbreviations that can be used for the term.
- Example (optional): Exemplary sentence that makes use of the term and its usual context.
- Status (optional): The status defines if the term is already "approved" by all parties of "proposed".
- Author: The author indicates the customer-side or developer-side individual who is responsible for the term.
- Business domain (optional): Some terms are only used within specific domains, e.g. within specific departments of a customer, while other departments make use of synonyms.

Syntax. The business glossary is defined within a table capturing for each of the defined terms within a glossary entry.

3.5. Artefact Type: Requirements Specification

The requirements specification encompasses all demanded properties of a (business information) system and / or a development process that shall be accomplished by a project in order to satisfy the demands stated in the business specification. Figure 3.16 gives an overview on the hierarchical structure of the requirements specification. It consists of following major content items:

- a *system vision* in content item 3.5.2: "What is the main scope of the information system in order to support a selected scope of the business processes landscape?".
- *information system requirements* in content item 3.5.3: "What properties shall the information system (including aspects of the application and architecture) have?".
- a *requirements risk status report* in content item 3.5.6: "What risks arise with the requirements?".
- *integrational requirements* in content item 3.5.4: "How shall the information system be delivered and deployed?".
- *organisational requirements* in content item 3.5.5: "How is the project constrained in its process?".
- a *glossary* in content item 3.5.7: "What are requirements-related (also technical) terms?".

Before defining the content of the requirements specification, we give a brief overview on the concepts of the different requirements classes, their attributes and their basic relationships.

3.5.1. Overview over Basic Requirements Types and Attributes

We conceptually classify for methodological reasons only those aspects as requirements that directly concern properties of information systems, their integration and their development process. Hence, in the context of this report, we define a requirement as follows:

Definition: ***Requirement***

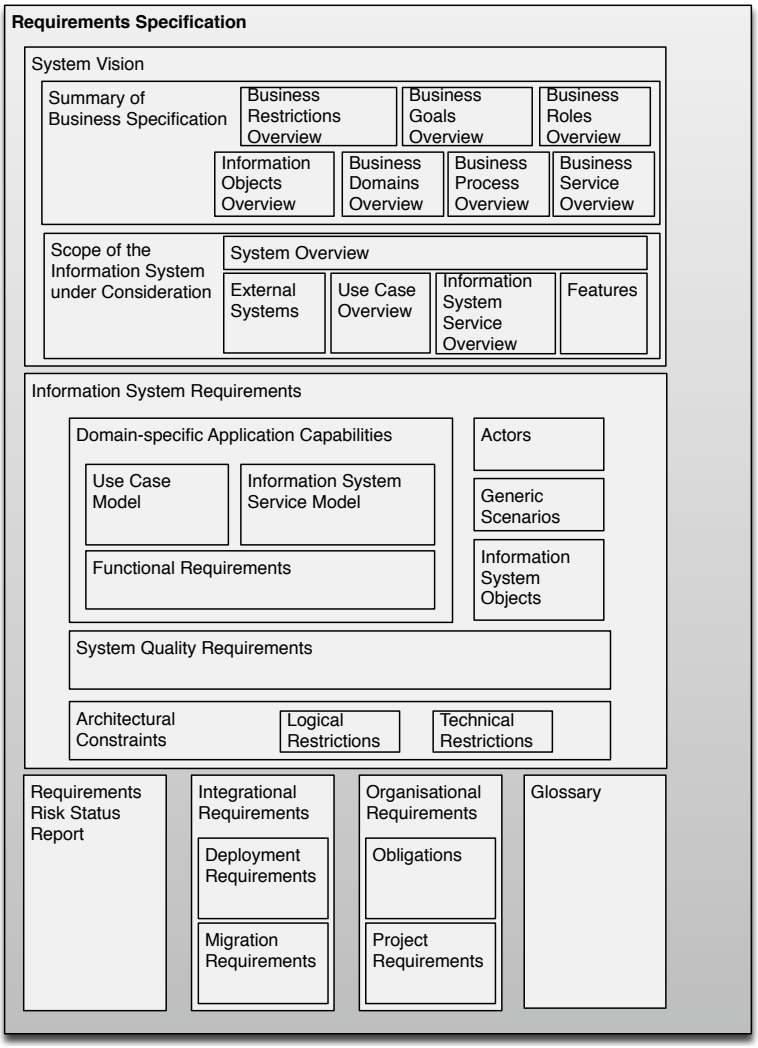


Figure 3.16.: Overview of Content Items of the Requirements Specification

A *Requirement* is a demanded property of an information system or a process and environment related to the system's development and integration, both to be accomplished by a development project.

Figure 3.17 illustrates resulting requirements classes and their attributes as found (besides other concepts) in the requirements specification.

According to this view we give a brief overview on the different requirements concepts and requirements attributes.

Requirements and Classification

We distinguish between three major content items that group several requirements classes according to their purpose.

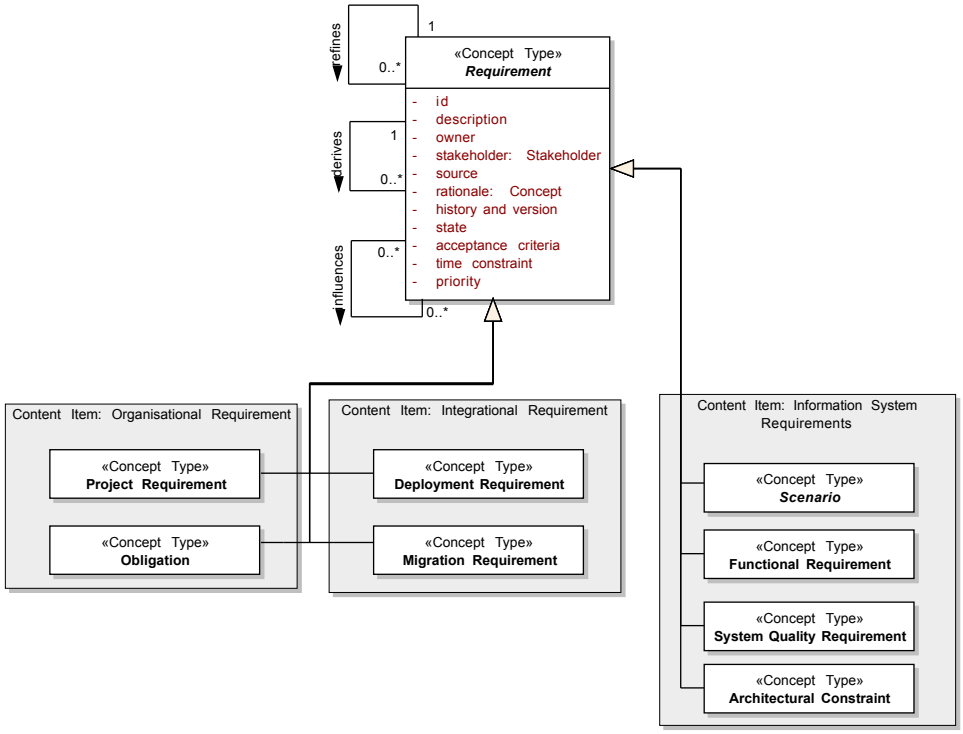


Figure 3.17.: Requirements Classification and Attributes

Information System Requirements. Information system requirements are requirements that exclusively demand specific properties of the application, the architecture and the (technical) environment of a particular information system. We distinguish between:

- *Scenarios* describing the demanded overall external behaviour of the application ("How will the system be used?"). We make use of scenarios for describing both (1) functional properties as part of a use case model in content item 3.5.3.3.1 and (2) non-functional properties as part of generic scenarios in content item 3.5.3.2.
- *Functional requirements* describing the demanded functional behaviour of the application ("What shall the system do in reaction to a specific stimulus?"). They construct the behaviour by describing specific functions an application must exhibit without defining how it must be realised.
- *System quality requirements* describing demanded properties of an application, its architecture and its environment ("What characteristics shall the system exhibit during behaviour and what further properties are necessary?"). They constrain the properties of the application and its environment by the use of specific measurements.
- *Architectural constraints* describing logical and technical restrictions on the solution of the overall system's architecture ("What is to be taken as a restriction during design activities?"). They directly restrict the "how", as they demand specific aspects of the architecture.

Organisational Requirements. Organisational requirements are requirements that restrict the overall project execution in its process. We distinguish between:

- *Project Requirements* describing restrictions on the overall process model ("What has to be done in what way?").
- *Obligations* describing additional agreements, propositions and exclusions between the parties within a development project ("Who has to do it? What will be excluded?").

Integrational Requirements. Integrational requirements are requirements that restrict the execution of the integration and the delivery of the system in its process and in its technical details. We distinguish between:

- *Migration Requirements* describing restrictions on the replacement or modification of one or more legacy applications.
- *Deployment Requirements* describing necessary details of the integration process from a customer's point of view. They restrict the technical environment during initial launch, but also corresponding organisational aspects ("What has to be delivered in what way?").

Requirements Attributes

Requirements attributes operationalise the requirements by defining relevant context-information (see also the introductory figure 3.17). A requirement without an attribute is neither controllable, nor is it traceable [RR99, RR07, Rup07]. The following attributes can to be taken as a recommendation. Variations can be made, depending on the project complexity.

Identifier (ID). Every Requirement has an unique identifier, as this is used to communicate with customers, testers, etc. How this ID is defined, depends on facts like the tool-support. We recommend to choose either a numerical value for the ID or in case of functional requirements a name of a function that they describe or affect (or a combination of both).

Description. The description covers the assertion of a requirement itself that is used as a basis to transfer a requirement into corresponding classes, enabling the use of specific concepts.

Owner. The owner of a requirement is the person who takes the responsibility in decision taking in terms of applying the requirement or of denying it.

Stakeholder. Stakeholders are one or more customer-side groups of interest that formulate requirements (see also content item business roles in 3.4.6.

Rationale. A requirement without a rationale can be a cost-intensive assumption perhaps without any value to the customer. The rationale justifies the existence of the requirement and its realisation. The rationale of a requirement is given by the concepts from which the requirement is derived (e.g. the rationale of a use case can be a specific business task).

Source. For the cases in which requirements cannot be rationaled by a specific concept, (informal) source of elicitation can be taken as an alternative. Examples for such sources are workshop protocols, emails or protocols of phone calls. It is advisable to document every source for a requirement (independent of if a rationale has been stated), especially within complex projects that have a large number of stakeholders and a high degree of communication.

History and Version. Every requirement underlies changes, e.g. due to resolved conflicts. The number of changes can be taken as an indicator for the stability of a requirement, which is for example important for planing and developing test cases. The higher the frequency of changes, the lower the stability. The history, however, defines a chronological view onto the performed changes and benefits an understanding of the requirements' life cycles respecting changes and refinements. With each of the changes of a requirement's content it changes its version, i.e. its current state by the use of e.g. a time stamp. Hence, the history covers a view onto all the versions, but still needs special attention for e.g. identifying moving targets within a project. How to define the history strongly depends on aspects, like the use of tools.

Hint: *Historising & Versioning Requirements by the use of Tools*
If using (requirements management) tools in order to administrate requirements, the differentiation between the history and the version of requirements is often blurred. Within white papers of different tools, both features are mixed up and solved by the use of e.g. subversion. Usually the version of a requirement should be captured in addition, while the history usually is reflected by the composition-tree in which requirements usually are visualised. If not referring to tool support, the attributes can be captured by a combination of both using the version as a prefix, followed by the last change and its date within the history (or all changes), e.g. "V.1.1 — 01.12.2008: requirements has been changed due to workshop protocol XY"

State. This attribute defines the current state of a requirement's life cycle. Figure 3.18 defines a possible set of states, also indicating possible transitions between the states that arise from mentioned life cycle.

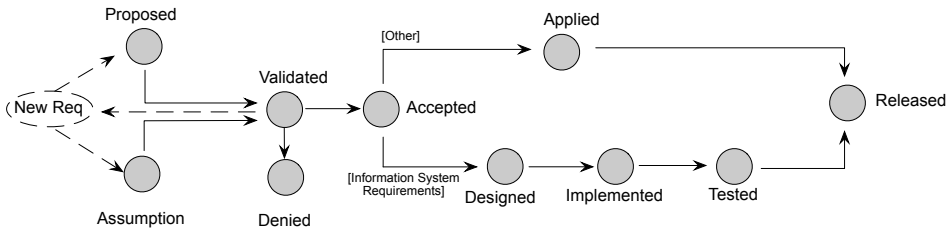


Figure 3.18.: States of Requirements

Acceptance Criteria. The acceptance criteria defines one or more circumstances under which a customer accepts to a solution for the corresponding requirement. This attribute supports the testability of a requirement. It is a measurement to enables the decision on if a solution matches the customer's expectations and can be compared to the *fit criterion* of the VOLERE Requirements Shell [RR07]. In general, requirements are often testable by nature (e.g. regarding functional requirements). But not all requirements are testable that easy and this is especially true if requirements remain underspecified (see also Sect. 3.2.3.3).

The acceptance criteria can be stated in two ways:

1. by defining a test case with an expected result
2. by defining possible technical solutions to which a customer might agree

Hint: *'Acceptance Criteria' versus 'Failure Classification'*
Besides the acceptance criteria, often a failure classification is defined within a project. The failure classification defines under which circumstances a requirement

is declared to ”failed to be realised” and especially the consequences, such as arranged penalties. The failure classification is part of project management and should be stated exclusively in contractual documents and not as part of the acceptance criteria.

Time Constraint. Time constraints allow the definition of a point in time in which requirements should be realised. Especially within time-boxing it is important to know the key requirements to be implemented for a specific release. We recommend delimiting this information to releases. Any other time-based information is part of project management. Therefore, one possible classification that allows such a flexible rating is: *High* (Before the next regular release, usually triggered by a change request), *Release X*, *Low* (may be postponed to further releases).

Priority. The priority of requirements describes the importance indicating the impact that requirements take for the customer and for the development if realising it or not. However, stakeholders will usually do hard in prioritising their requirements by themselves for example as “low”, simply because it is in human nature to see his own requirements all as of high priority and therefore as a ”Must” [5]. In complex projects with a high degree of communication, or with an unknown customer, it is nearly impossible to resolve this problem. Therefore, one possibility is to prioritise the requirement in a reasonable way without too many individual influences of the stakeholders [Wie03, Rup07] following a cost-value-based approach [KR97].

Within the business demands analysis (in content item 3.4.7) we described the needs regarding each business service in terms of the costs that are implied, the business value (importance to the customers) and finally the business risks, i.e. the risks to the customer’s organisations. As the requirements are derived from the business services, the priority of the requirements can also be derived from the business demands regarding envisioned business services. To capture also the development aspects within the prioritisation of the requirements one should extend the business demands with the risks that arise from the requirements (see also content item “Requirements Risk Status Report” in 3.5.6), e.g risks that arise from quality requirements that are technically not feasible

3.5.2. System Vision

The system vision defines the sometimes contractual basis for the requirements specification. It defines how one particular information system will reflect the needs of the business, respectively business processes (“what business processes shall be supported by the system?”). It builds therefore the core interface between a requirements specification and a business specification. It defines 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. The following detailed information system requirements then define in content item 3.5.3 the demands in their depth (“to which degree shall the business processes be automatised and how exactly?”). The system vision therefore shows how the system will match into his intended future “real world” - environment, given by the business process and system landscape of a customer’s organisation. This is done by mapping an external behavioural view of the application onto the content of the business specification.

5 This is also the reason why we take a critical stance towards prioritisation methods like “MuSCoW” (“Must”, “Should”, “Could”, “Would”).

For this purpose, the system vision consists of two major content items that define the scope of the development project in terms of:

- summarising the essential parts of the business specification that are in scope of one particular system (Summary of Business Specification)
- defining and matching high-level functionality of this system with the scope of the business specification (Scope of the Information System under Consideration)

The benefit of the system vision consists of scoping and detection of moving targets. It aligns all stakeholders into a common direction as it initially defines "what the information system shall basically become". Hence, it decreases the possibility of scope creeps and by being aligned to the business specification it supports that following requirements will satisfy business goals. Furthermore, it builds the basis for project management activities respecting estimations of complexity and effort and respecting the definition of contractual agreements concerning development projects (see also section 3.8).

Figure 3.19 illustrates the concepts of the system vision. The upper part within the content item describes the concepts used to summarise the business specification, the bottom part describes the concepts used to define the scope of the information system. On the left side of the figure, the requirements concept is depicted as an abstract class. The already introduced requirements classes are not part of the system vision, but part of subsequent content items.

For reasons of complexity we do not depict all dependencies within the concept types of the summary of the business specification. In particular, we do not depict the dependencies from the business services to the business processes that provide the service and to the stakeholders that (potentially) call the services.

3.5.2.1. Summary of Business Specification

The summary of the business specification describes those aspects of the business that are in scope of the envisioned information system. As we make use of the same concepts of the business specification, but (if at all) differ only in the representation of the like, we only give a brief description of the content items:

Business Goals Overview. The summary of the business goals includes those goals that affect the envisioned information system, in particular emphasising the IT-related goals for the purpose of defining system quality requirements (in content item 3.5.3.5)[6].

Business Restrictions Overview. The summary of the business restrictions includes in particular the depiction of

- those business rules that serve as an input for the elaboration of scenarios, e.g. defined as part of the use case model in content item 3.5.3.3.1 and as part of the generic scenarios in content item 3.5.3.2.
- those business constraints that serve as an input for the elaboration of architectural constraints in content item 3.5.3.6 and information system objects in content item 3.5.3.4 (concerning e.g. the enforcement of specific naming conventions).

6 The process-related goals serve as an input for the elaboration of the use cases in content item 3.5.3.3.1 but are implicitly covered by already (goal-compliant) business task descriptions that are used as a basis for elaborating the use cases.

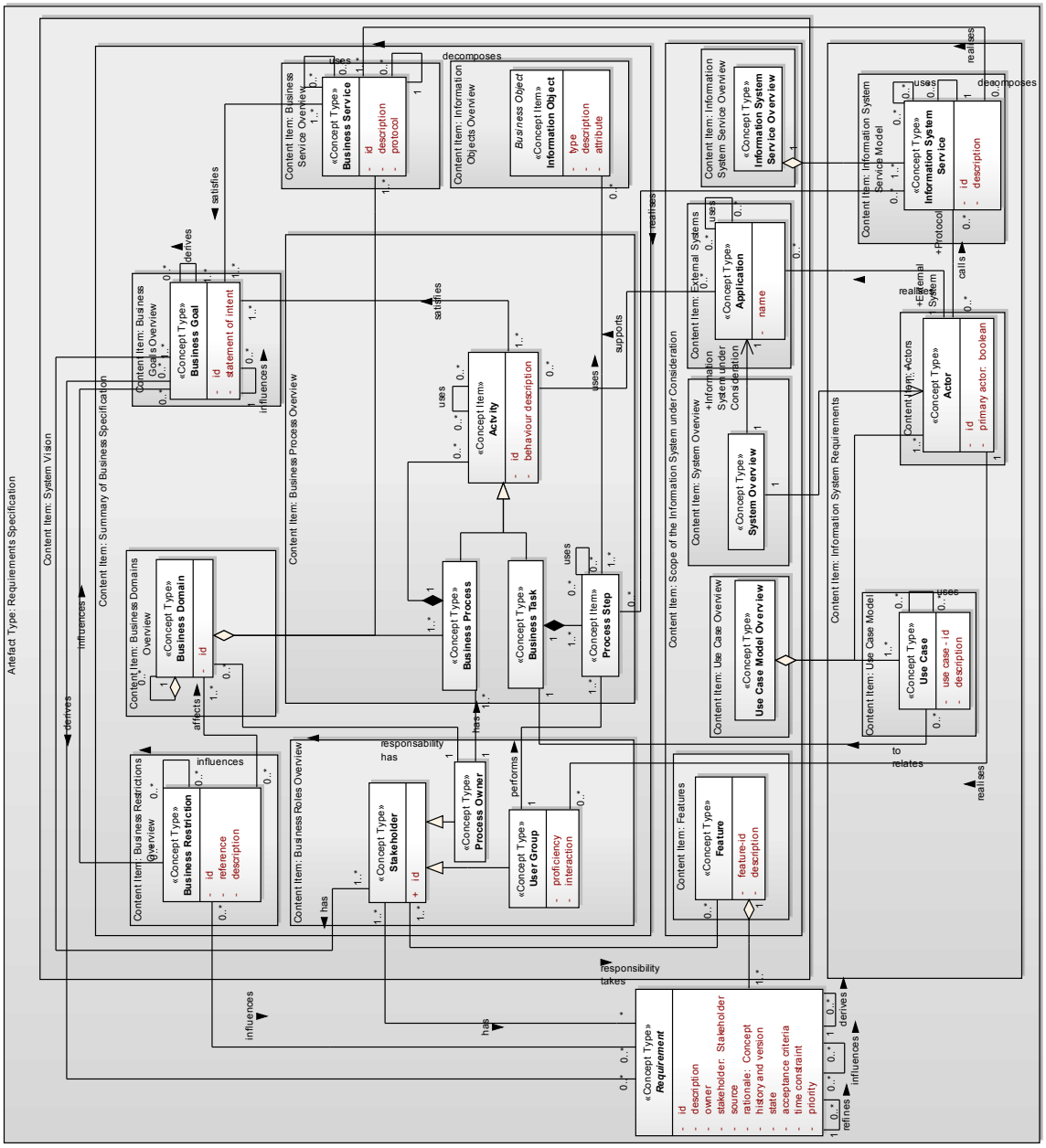


Figure 3.19.: Overview on *System Vision* and underlying Concepts

Business Roles Overview. This summary emphasises the user groups that shall be supported in their business tasks by the envisioned system. The description of the user groups guides the definition of the actors in content item 3.5.3.1.

Business Process Overview and Business Service Overview. Both content items serve as an input for defining domain-specific application capabilities (functionality), in particular for defining the functional behaviour with means of e.g. a use case model.

Business Domains Overview. The business domains that are in scope of the information system serve as a means to (functionally) structure the application, compliant to the functional structure of the business process landscape. This structure extends from the structure of the use case overview and the system overview (content items 3.5.2.2.1 and 3.5.2.2.3) to the structure of the application capabilities in content item 3.5.3.3 beyond to the initial structure of the architecture into components.

Information Objects Overview. This summary, finally, serves as a basis for defining information system objects in content item 3.5.3.4 — a system-side representation of the information objects.

3.5.2.2. Scope of the Information System under Consideration

The information system under consideration is initially defined from an external (behavioural) perspective. This is done by defining:

- Use case overview: The use case overview lists the major use cases that are directly related to the business tasks. This overview defines what functionality of the application will support the business processes — what processes will be (at least partially) automatised.
- Information system service overview: Information system services describe application's functionality in terms of a logical representation of functions that can be called by actors.
- System overview: The system overview defines the system's boundary and how it interrelates with its future environment (its context).
- External systems: This concept summarises the identified surrounding systems that interact with the envisioned one.
- Features: The features additionally define user-visible characteristics of the system that group a specific set of requirements.

3.5.2.2.1 Use Case Overview

The use case overview summarises the use cases in relation to the business tasks. The use cases within the overview provide a quick point of entry by depicting how functionality of the system shall support the business in realising its idealised activities. The use cases are identified (and listed), but specified in full with corresponding scenarios ("How shall the business tasks be exactly supported") within the use case model (see also content item 3.5.3.3.1).

Definition: ***Use Case Overview***

The *use case overview* 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. It defines the relationships between actors and use cases as well as between the use cases.

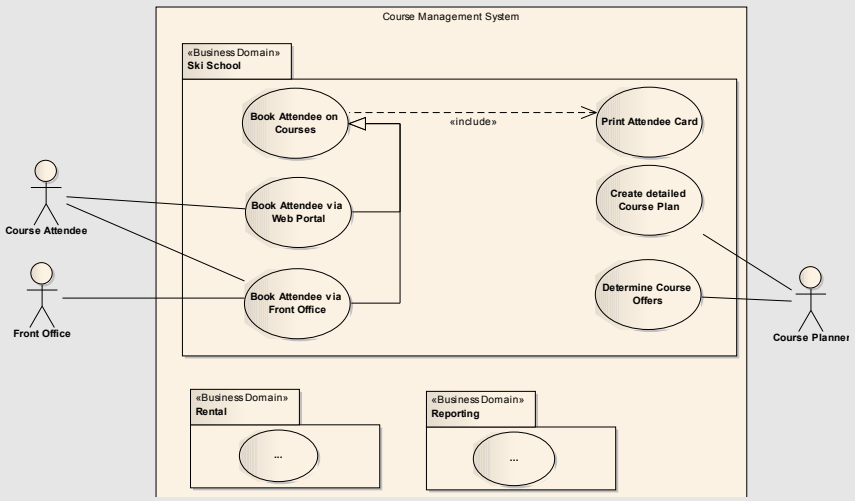
Content Dependencies. The use cases within the overview can and should be structured according to the business domains. For each of the business tasks that has to be automatised in some way (within these business domains), we can identify one or more use case as the business tasks serve as candidates. The use cases then define with specific scenarios how single actors shall be able to interact with the application in order to perform their business tasks. Obviously, the use cases

can still be arranged differently as the business tasks (i.e. by representing one business task by several use cases). Still, one use case should never relate to different business tasks since a use case cannot describe the system-side realisation of business logic in the granularity of more then one business task [Coc00].

Finally, the dependencies in-between single use cases (the "uses"-associations) are to be set within the overview, although these usually will be known when the use case model has been elaborated in full (see also content item 3.5.3.3.1).

Syntax. We recommend to describe the use case model overview with a use case diagram as proposed by the UML [OMG07a, OMG07b]. It is depicted by example 3.6.

Example 3.6. *Use Case Overview*



This diagram provides a quick overview of the use cases with their dependencies to other use cases and to the actors that execute the use cases. Within the use case diagram, however, the use cases are assigned to the business domains, e.g. by the use of packages (one package representing one business domain). The diagram then provides a clear graphical representation of the relationships between the actors and (business) domain-specific functionality that is offered by the application.

3.5.2.2.2 Information System Service Overview

The information system service overview summarises all the information system services that are offered by the envisioned system and that result from the necessity of supporting the business processes, respectively business services. Note that similar to business services and their relation to the business processes, an information system service describes a logical representation of the work that is performed with the support of the information system, i.e. an abstract view onto a use case.

However, the concept of the information system service is described in detail within the information system service model in content item 3.5.3.3.3.

Content Dependencies. The only dependency is given by the information system services that are referred by the overview (at least one).

Syntax. Similar to the business service overview, we recommend to summarise the information system services in a list, because a detailed representation of the single information system services (including their dependencies in terms of collaboration) is given in content item 3.5.3.3.3. This list should capture at least the ID of the information system service and optionally the business service that is realised by the information system service.

3.5.2.2.3 System Overview

The system overview defines the system's boundary and is conceptually defined as follows:

Definition: ***System Overview***

The *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.

In contrast to the use case overview in content item 3.5.2.2.1, the system overview emphasises the operative environment and its dependencies to the envisioned system. The use case model in turn emphasises the application and what functionality it shall offer to its environment, what is depicted by the use cases. Similar to the use case overview, the system overview should be structured according to the business domains by grouping the external actors that interact with the system according to the business domains which group in turn the business processes they are related to.

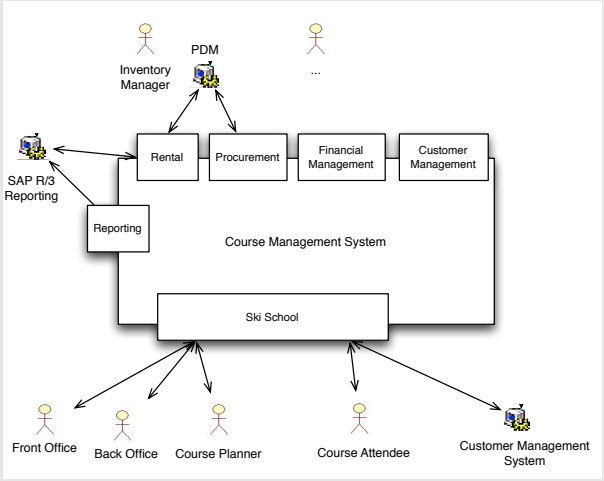
The information to be defined within the system overview is therefore:

- Business domains: The business domains that structure the interrelating Actors.
- Actors: User groups and / or external systems that communicate with the application for each of the defined business domains (see also content item 3.5.3.1).
- Interrelation: This information defines how the actors intent to communicate with the application and indicate the direction of the information flow (if unidirectional).

Content Dependencies. The system overview describes the interrelation of the system with its environment that consists of either external systems (see content item 3.5.2.2.4) or user groups (see content item 3.4.6.3). As both are described by means of actors in content item 3.5.3.1), the system overview refers at least to one actor that interrelates to the system.

Syntax. We recommend to describe the system overview by first giving a diagrammatic overview that depicts the operative environment of the system within a context diagram, as depicted by example 3.7. This diagram structures the environment according to the business domains and depicts which user groups and external systems will directly communicate with the system within each of these domains and finally how they will interrelate (depicted by the arrows). The arrows also indicate to which direction the communication is performed. The direction of the communication gains also as an input to define primary and secondary actors.

Example 3.7. *System Overview (Context Diagram)*



For each of the depicted business domains that are directly related to the system, a table can be used to depict the actors and describing how they communicate with the system.

3.5.2.2.4 External Systems

Within the system vision, we generally define what is in scope of the envisioned system and what is out of scope. Regarding the latter, we still define for the purpose of a proper overview (especially regarding application landscapes) what business tasks shall be supported by other systems and how they (directly and indirectly) relate to the envisioned one as a consequence of the workflow description within the business process model.

For example, regarding the business process "Order Travel", the support of the envisioned system might consist of several use cases, such as "Search Flight", "Search Hotel", etc., while one aspect within the workflow, such as accounting, could be performed on another system. This system does not directly interact with the envisioned one interchanging information, but is referred by a user in order to perform his business tasks, thus, the information is still necessary. Furthermore, information that enable the depicted use cases "Search Flight" could be provided by external systems, while these in turn would directly interact with the envisioned one.

Hence, we conceptually define external systems as follows:

Definition: ***External System***

An *external system* is a system including one or more applications that directly or indirectly interacts with the envisioned system in order to realise the workflow of a business process.

Content Dependencies. The definition of the external systems (that directly interact with the envisioned one) serves as an input for the elaboration of the actors in content item 3.5.3.1, and is referred as a consequence by the system overview.

Syntax. There is no universal way to represent the external systems. We recommend to describe the systems textually within a list, defining a name, optionally the applications that they provide (e.g. SAP) and finally the business processes that they support, respectively to which they are related.

3.5.2.2.5 Features

Features (and their notion) are heavily discussed and often taken as a specific type of requirements. In fact, features mostly arise from the area of product lines development, in which features are stated and decomposed to trees in order to illustrate possible variants of same and similar product characteristics. Consequently, there exist many ways of expressing and understanding features. Within the application domain of business information systems, the concept of features is not frequently used but they still can be stated by customers e.g. within the telecommunication sector in which features are often used for billing the use of several functions under one specific term to a market.

In the context of this report, features are not requirements in the nearer sense. They just are recognisable characteristics of the information system that are important to a customer and that define a specific view onto a set of detailed requirements. For instance, characteristics in terms of quality characteristics as they are represented by the ISO Std. 9126 [ISO03]. Or technical characteristics. As these demands are already covered by the different requirement classes and corresponding concepts, we declare the features according to Wiegers as a set of logically related requirements that provide a capability to user groups. Features are recognisable to a stakeholder that aids in making a purchase decision — a bullet item in a product description [Wie03]. Features serve therefore as a means to bundle several requirements under one specific characteristic.

According to [CHS08, KCH+90], we use the following definition for features:

Definition: *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 that fulfils this feature.

Features are therefore means to group a specific set of requirements under one (for a stakeholder) memorable substantive, such as "Transaction Safety", "Revision Safety" or "Automatic Tax Calculation". These substantives define specific views under which same and similar requirements can be discussed and tackled from different angles.

For example, one stakeholder could demand the feature "Transparency". Within the logistics sector this feature includes tracking and tracing shipments at the basis of single packages, instead of on the basis of whole palettes. This feature then groups a set of use cases in which the tracking and tracing for the shipment processes is described.

As feature define a selection of specific requirements and support the communication of same or similar requirements to different stakeholders, the concept of the features only consists of the following content:

- Feature-ID: A unique identifier
- Description: A short description of the feature
- Stakeholder: the stakeholder that is interested in this feature
- Referenced Requirements: defines the requirements (IDs) the are grouped by this feature, that satisfy this feature.

Content Dependencies. The dependencies of a feature are given by at least one requirement that is grouped by the feature and at least one stakeholder that takes the responsibility of the feature.

Syntax. Features should be specified textually within a list or within a spreadsheet. Note that compared to other feature-related approaches (e.g. found in the application domain of embedded reactive systems), they must according to the definition not be documented hierarchically.

3.5.3. Information System Requirements

Information system requirements are requirements that exclusively describe specific demanded functionalities, properties and conditions of the application, its architecture and its environment from a customer's perspective. We distinguish between several content items in which we state for reasons of complexity for each the encompassed excerpt of the concept model. In particular, we define within the information system requirements:

- *Actors* in content item 3.5.3.1 as a representation of all the external entities that interact with the future system, reflecting the user groups and the external systems.
- *Generic scenarios* in content item 3.5.3.2 as means to describe high-level non-functional properties that crosscut several business domains.
- *Domain-specific application capabilities* in content item 3.5.3.3 defining functionality — clustered into business domains — that is necessary to realise a seamless (functional) support of the business process logic for the single business domains. We distinguish in particular between three further content items:
  - *Use case model*: in content item 3.5.3.3.1 describing sequences of interaction (scenarios) between actors and the system as a whole illustrating how the system will be used in the context of the business processes.
  - *Functional requirements* in content item 3.5.3.3.2 defining single demanded functional properties of the application in order to enable the defined use cases.
  - *Information system services* in content item 3.5.3.3.3 defining a logical abstraction of the use cases (without defining scenarios or necessarily involving actors).
- *Information system objects* in content item 3.5.3.4 reflecting the information objects (of the business process logic) to be processed by the information system as a consequence of defined functionality.
- *System quality requirements* in content item 3.5.3.5 constraining behavioural and structural characteristics and properties of the system and its environment.
- *Architectural constraints* in content item 3.5.3.6 restricting the future architecture by defining:
  - *Logical restrictions* in content item 3.5.3.6.1
  - *Technical restrictions* in content item 3.5.3.6.2

3.5.3.1. Actors

Figure 3.20 illustrates the concept for describing actors, situated on the right side of the figure. On the upper part of the figure, we depict the concepts from which

the actors are derived (that they "realise"), i.e. the user groups from content item 3.4.6.3 and the external systems from content item 3.5.2.2.4 both described within the system vision.

Actors, however, are a logical representation these external entities that intent to to participate in a scenario (e.g. a use case) or may call an information system service. Each defined actor exclusively consists of a unique *id* ,a *reference* that indicates to the source from which is has been derived and finally if it is a primary or a secondary actor, i.e. if the actor communicates bidirectionally or unidirectionally only using information as an input (see also the system overview in content item 3.5.2.2.3).

We conceptually define actors as follows:

Definition: *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. We distinguish between primary actors that provide and use information and secondary actors that only use information from the system as an input.

Note that it is possible to group several user groups or systems by one actor. In particular, user groups are defined and structured by their characteristics within the customer's organisation (their abilities, their needs and their relation to certain business domains). Instead, actors are defined and structured by the functions of a system that they can execute (e.g. the use cases or the services). Hence, if two different user groups can execute for example same and similar use cases, both should be grouped to one actor.

Content Dependencies. Actors can call one or more information system services and participate in at least one scenarios, e.g. as part of a use case. They can be related to either one or more user groups or to one or more external applications.

Syntax. There is no universal way of describing actors. We recommend to textually list them within an actors-list. When documenting the actors in such a list, it should capture for each actor the *reference* to its source and finally the information if it is a primary actor or a secondary one.

3.5.3.2. Generic Scenarios

Figure 3.21 on page 87 illustrates the concept for describing generic scenarios, situated on the left side of the figure. Scenarios in general do not make any assertion on a requirement's type. They just state specific interactions with the application and conceal several quantified requirements. Usually, the common understanding on scenarios within RE is that this concept is meant to illustrate functional requirements.

In fact, it depends on what kind of scenario we use, thereby the scenario is described as an abstract class. A scenario can either focus on:

1. **Functional properties:** The scenario is described in this case as a structured sequence of events and responses, being part of a use case in order to specify functional properties of the application (e.g. "How will a certain user group use the application in order to perform the business task 'book flight'?"). This kind of scenario is a functionally motivated *structured scenario*, in in the figure depicted as part of a use case.

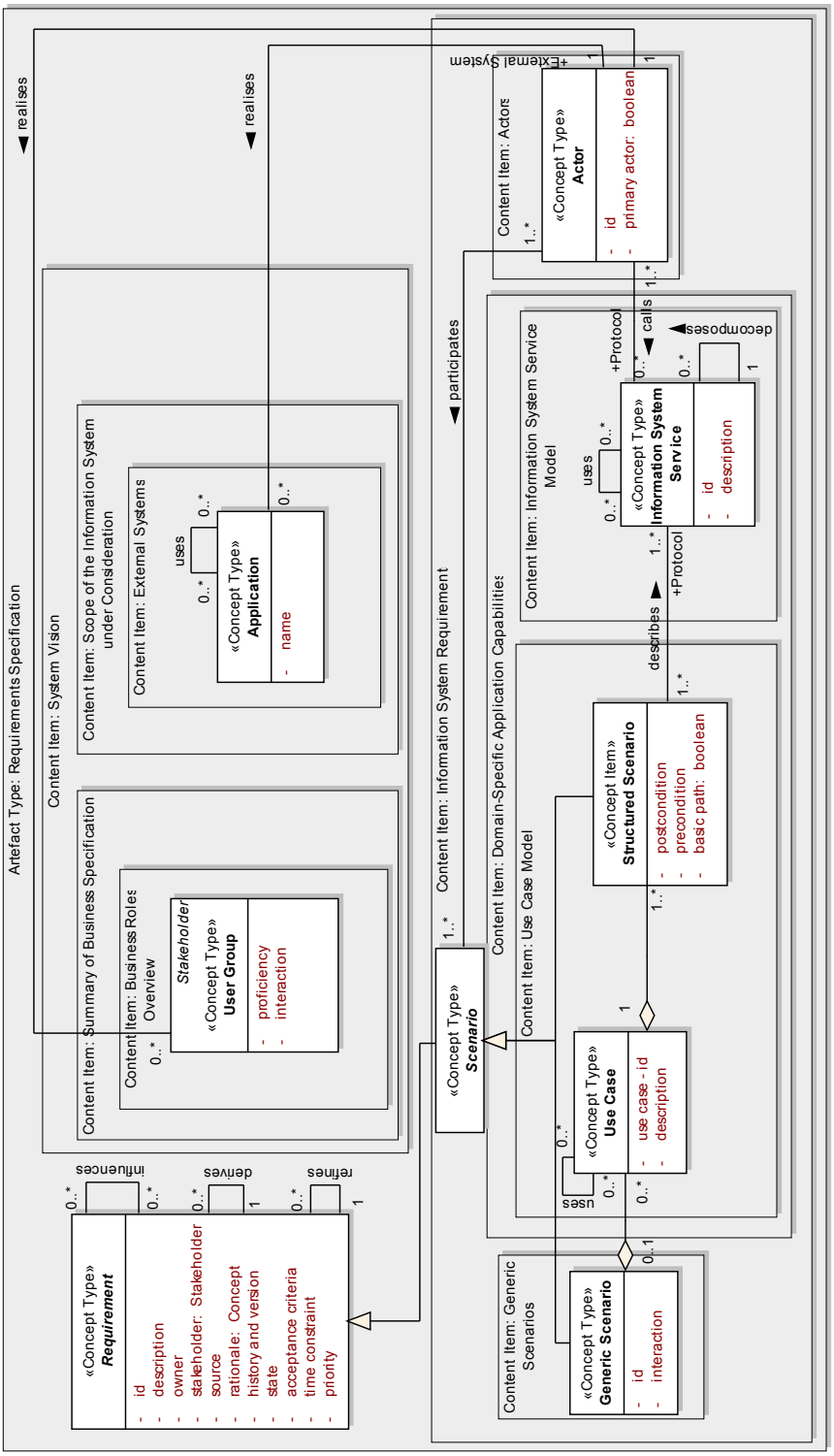


Figure 3.20.: Overview on *Actors* and underlying Concepts

2. **Non-functional properties:** The scenario is described in this case generically in order to specify quality and architectural demands that are not directly

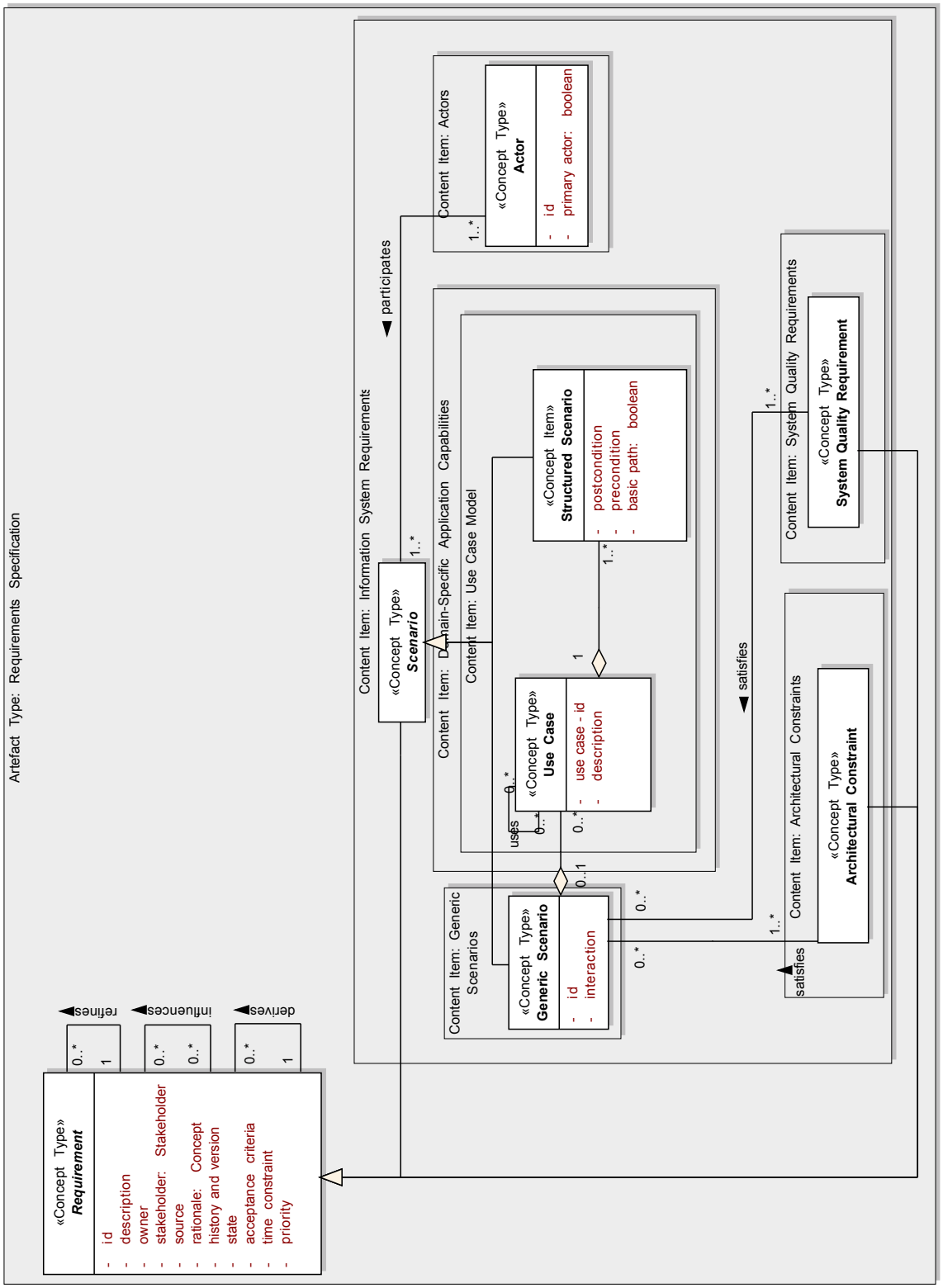


Figure 3.21.: Overview on *Generic Scenarios* and underlying Concepts

related to the application's logic (e.g. "How will an administrator perform updates of the system during maintenance activities?"). It is not part of a regular use case description and referred to as a *generic scenario*, in the figure depicted on the left side.

A generic scenario, however, describes non-functional aspects that are not necessarily related to any kind of business task. It is any imaginable sequence of steps that an actor can perform on the application as a whole, the system's environment, or on specific structural elements of the system. For example, the modification of the code, the web browser or the documentation. It has a non-functional character, because it does not illustrate how the functionality of the application directly supports a user group in performing his business task. No functions can be directly called. Hence, compared to structured scenarios within use cases, we do not explicitly define information system objects that are processed, nor do we have a direct relation to business tasks.

In figure 3.21, the non-functional purpose is represented by the *satisfies*-association between one generic scenario and many system quality requirements and / or architectural constraints that can be derived from such a scenario (see also content items 3.5.3.5 and 3.5.3.6).

The use of generic scenarios benefits:

- the description of requirements that are hard to specify in an unambiguous and testable way without directly having to quantify them. Instead of having to think on e.g. precise system quality requirements that aim for example at a certain maintainability, one can describe specific maintenance activities that an administrator will have to perform on the system. They help in initially understanding how the application will be used, especially if a customer is not able to precisely state his exact needs and expectations regarding quality aspects.
- the estimation of (functional) complexity and thereby efforts respecting quality or architectural aspects (performing sizing analyses).

Depending on the assertion we want to make with a generic scenario, we differ between three kinds of generic scenarios and define the concept as follows:

Definition: ***Generic Scenario***

A generic scenario is an abstract description of an interaction between an actor and the system or its environment with the purpose of:

1. describing interactions / behaviour, that has to be *prevented* (by the means of attack scenarios)
2. describing interactions / behaviour, that directly has to be *supported* without any relation to the business process logic (e.g. by describing maintenance tasks that in general address the internal quality of an application)
3. structuring, ordering and grouping use cases (emphasising what users have to perform *before executing* a use case or *between executing* several use cases

---

In the first case (of the definition), we refer to scenarios as they are defined as part of Misuse Cases (see also [HP08, Ale03]). They illustrate possible actions that attackers can perform on the application and that have to be prevented (see also [WMFIL09], where we explicitly illustrated the use of such scenarios). In the second case, however, we refer to scenarios, as proposed by ATAM (Architecture Tradeoff Analysis Method), a method for evaluating architectures (see also [BCK05]). They illustrate interactions that e.g. the IT department will perform when for example modifying the code. These interactions are directly performed

on the system and its environment and are not necessarily related to the business process. In the third case, we refer to the story behind a use case that gives a big picture that is often not possible when exclusively describing use cases and business tasks. Example 3.8 illustrates the three possibilities of using generic scenarios.

Example 3.8. *Exemplary Generic Scenarios*

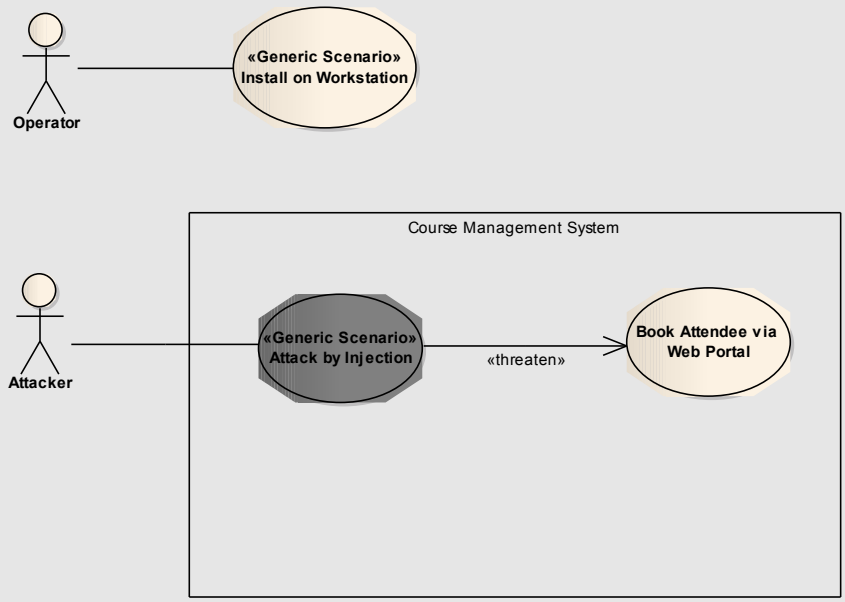
1. *Attack scenario illustrating how an attacker intends to embed malicious code in HTTP query strings and that has to be prevented: "1. Attacker exploits injection attacks; 2. Attacker exploits all public links on a web site using a spider tool; 3.: After detecting a vulnerable parameter, the attacker creates exploit URLs and gets victims to click on them; . . .*
2. *Administrative scenario illustrating an administrative task: "The administrator shall be able to modify the data scheme of the database during run-time without any side-effects in a maximum of three hours."*
3. *Scenario illustrating what a user has to do before executing a use case: "The bank employee shall be able to order a stock (regular use case) after reading the manual within a maximum of 3 hours".*

Generic scenarios offer a good means of discussing with stakeholders about their quality needs and gain as a basis for deriving further detailed requirements. Referring to the attack scenario of example 3.8 one possible derived system quality requirement could be "The software shall sufficiently sanitise user-controllable input for content before it is prepared in output that is used as a web page". Referring to the administrative scenario a derived system quality requirement could be "The documentation shall be written in English and summarise all necessary information of frequently accessed use cases as a quickstart guideline".

Content Dependencies. A generic scenario has no enforced relations. It can group one or more use cases and we can derive one or more system quality requirements or architectural requirements.

Syntax. Because of the different possibilities of expressing generic scenarios, the use of description techniques that illustrate control flows makes no sense. We recommend therefore to remain to an informal description of the interaction in natural language. For instance, as known from story cards that are used in agile development projects, while allocating to each of the described scenarios the requirements attributes. In addition to the description of the generic scenarios, we recommend to depict all relevant scenarios in an overview diagram that shows how the scenarios relate to (functional) use cases (see also the example).

Example 3.9. *Generic Scenarios (Overview Diagram)*



3.5.3.3. Domain-specific Application Capatilities

Within this content item we define functional properties of the application, specifically structured into business domains. Consequently, the content items within this one item can be reproduced within the information system requirements for each of the business domains (each chapter representing one particular business domain, see also appendix A.1.2).

For each of the business domains, we define (1) a use case model, (2) functional requirements that are related to these use cases and finally (3) information system services.

3.5.3.3.1 Use Case Model

A use case model in general groups a set of logically related use cases that can be performed by one or more actors. Use cases in turn are to be seen as a collection of related structured scenarios — a collection of interactions. These interactions can be performed between an actor and the application as a whole, but also between two components. This depends on the level of abstraction, in which we describe a use case. With the concept of a use case, however, we refer according to Cockburn [Coc00] to use case at "sea-level". This means to interactions between actors and the application as a whole (seen as a black box), performed in order to realise chosen business tasks within one particular business domain.

Figure 3.22 illustrates the concept of the use case model, its relation to functional requirements (on the bottom part of the figure) and especially its relation to the business tasks, from which the use cases are derived.

Each of the use cases is a means to understand the expectations of particular user groups and how these (and external systems) will interact with the application ("How will the business tasks be performed by the use of the information system

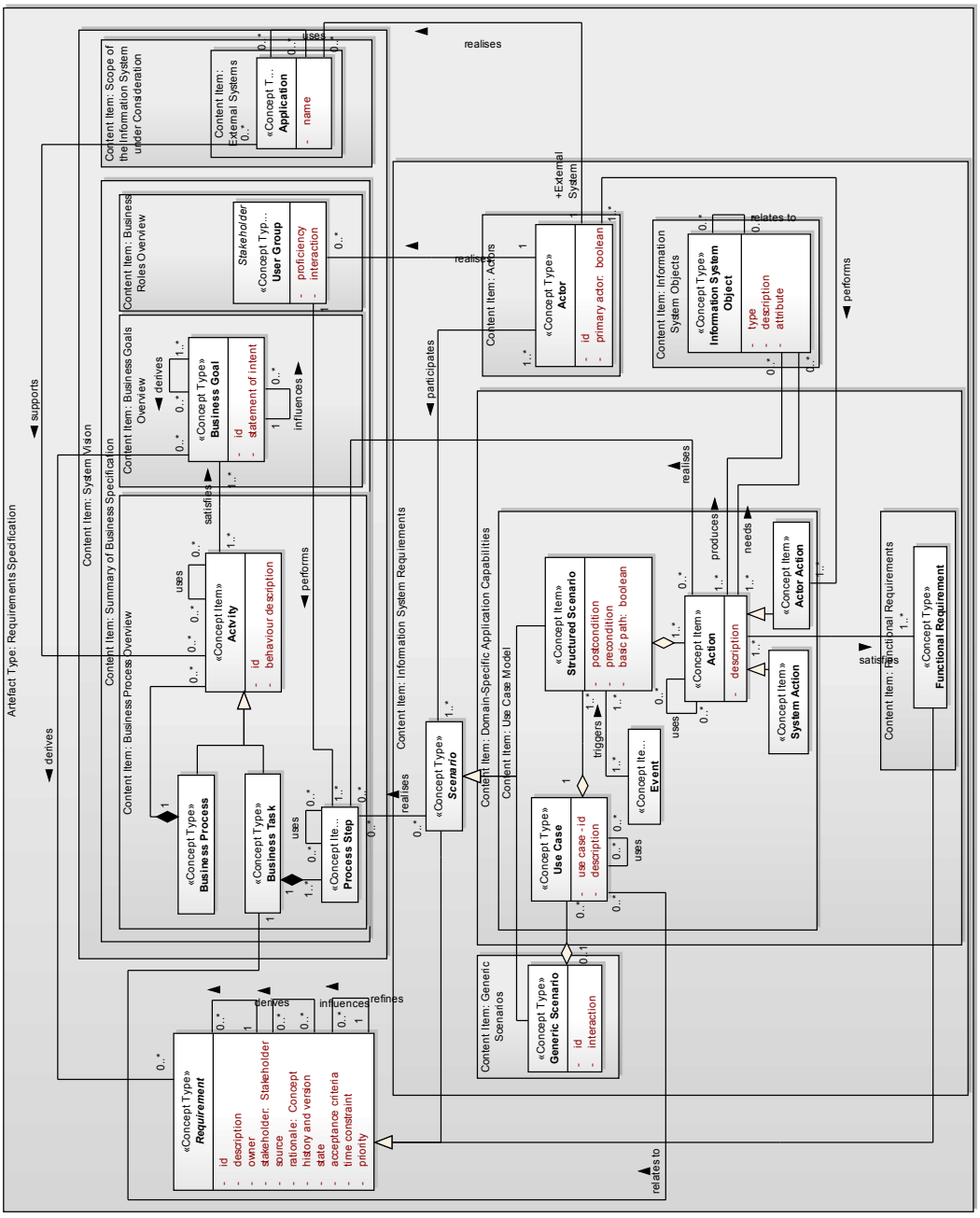


Figure 3.22.: Overview on *Use Case Model* and underlying Concepts

and what is expected as a result?"). They define what business tasks have to be supported in what way. Hence, use cases, (structured) scenarios and further related concepts are derived from the business process model that does not involve any aspects of the application. When deriving use cases in order to define how the business processes will be supported by the application, we refer in particular to the business tasks and included single process steps that the users will perform by their own and the steps that will be performed by the application.

A use case describes this support with a set of structured scenarios that are logi-

cally related to each other. For example, all possible interactions that can be performed within the business task "add trainer to course". Each of these scenarios is characterised by a basic path or an alternative path. A basic path describes a scenarios that is primarily expected by an actor. The regular one that is executed. For instance, what a user group "usually wants to do with the application when adding a trainer to a course". An alternative scenario describes what "may occasionally happen when adding a trainer to a course", such as cancelling the procedure.

When describing such a scenario within a use case, it consists of an ordered sequence of actions (an action therefore "uses" another action). These actions describe what exactly is performed by an (external) actor, representing a user group or an external system, or what is performed by the envisioned application. We consequently distinguish between actor actions and system actions. Actor actions define what the actors perform, while system actions define what the system shall do in order to enable the expected response.

Exactly these actions realise the behaviour of a business task. When referring to a business task, one automatically refers to a specific sequence of process steps that are performed within a business task. Therefore, the transition from the business process model to the analysis of information systems is performed by describing structured scenarios that define for each of the business tasks what of the included process steps are transferred to either an actor action or to a system action.

Within the concept model, this transition is defined by the association between the action element and the element of the process step.

Therefore, we conceptually define according to [Wie03, Coc00] a use cases as follows:

Definition: ***Use Case***

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 to satisfy a business goal. 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. It consists of a unique ID and a brief description of its purpose. A use case is related to:

- an actor that participates within a structured scenario.
- a structured scenario: A structured scenario is an ordered sequnce 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).
- Action: An action is 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 performed by an actor, a system action a process step that is realised by the application.
- Information System Object: An information system object is the system-side representation of an information object that is created or modified as a consequence of a system action.

Note that each scenario includes a condition rather than a whole use case including a condition, e.g. the precondition that the actor has to be logged on before being able to book a flight.

Hint: *Stating Preconditions and Postconditions in Scenarios instead of in Use Cases.* Often, the conditions are stated within the use cases and not within its single scenarios. If stating the conditions within a use case, these conditions affect all the

scenarios within the use case. The main advantage of stating the conditions for each scenario consists in the gained testability. When generating test cases from a use case, testers are interested in single scenarios within the use cases, not in the use case as a whole.

Finally, use cases and structured scenarios do not include any system quality requirements (e.g. performance-related ones). These are documented separately. See also the corresponding concept of system quality requirements in content item 3.5.3.5.

Content Dependencies. Use cases can be grouped by generic scenarios. Each of the defined use cases may relate to other use cases (that are "used") and to single business tasks that shall be supported (in parts) by the information system. Furthermore, a use case groups at least one structured scenario that is triggered by exactly one event. Each structured scenario in turn, in which at least one actor participates, consists of an ordered sequence of at least one action. Finally, as part of each action, one or more information system objects can be modified. Note that structured scenarios can be derived from business rules (actionable statements), but is for reasons of complexity not depicted in the figure (see also the system vision in content item 3.5.2).

Hint: *Structured Scenarios versus Generic Scenarios*
The difference between structured and generic scenarios is given by the associated concepts. Compared to generic scenarios, the structured ones consist of (functionally motivated) actions that are performed by either actors or the system. Furthermore, within structured scenarios, information system objects can be processed.

Syntax. There exist two possibilities of representing single use cases[7]:

1. Textually written in natural language, in which the use case-id, its description, the actors and finally the structured scenarios processing information system objects are written within templates as described in [Coc00].
2. By means of a diagrammatic representation in which the single scenarios including the actions and their dependencies to other actions are additionally represented covering all possible paths, i.e. basic and alternative ones.

Especially if representing complex interactions, in which a textual description would cover more then one page, we recommend according to [Wie03, Coc00] a diagrammatic representation (see also example 3.10).

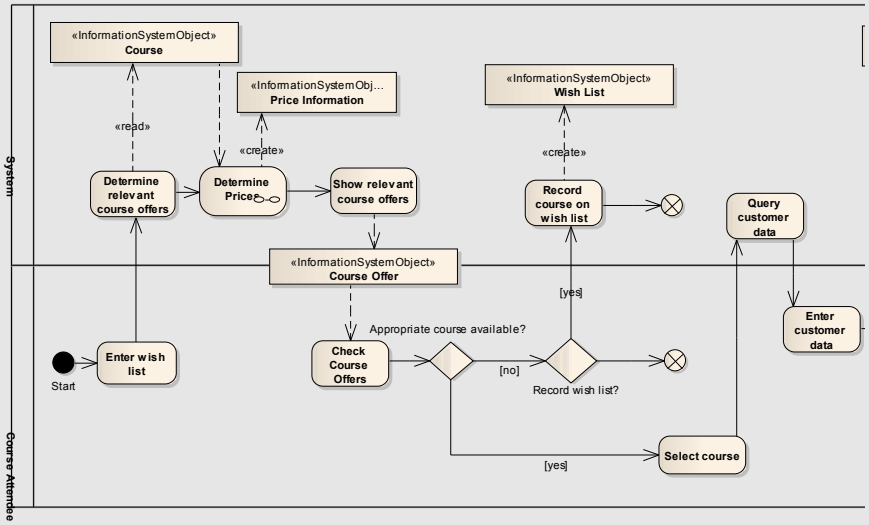
For such a representation, there exist, similar to the syntax of the business process model in content item 3.4.4.2, several possibilities of representing use cases. For the construction of scenarios with ordered sequences of actions and their relation to information system objects, a combination of data and control flow-oriented techniques are necessary. Furthermore, in order to explicitly distinguish between the activities that are performed by actors and ones that are performed by the envisioned application as a whole, the techniques must offer means for this purpose.

Although, behavioural diagram types with a particular focus on interactions like the established UML activity diagrams offer these concepts (distinguishing by the use of e.g. "swim lanes"), one methodological problem consists in a seamless transition between the use cases (at the abstraction level of the information system service hierarchy) to the behaviour description from an architectural (glass box / white box) point of view respecting the granularity of the use cases. Within this transition, the system actions of the use cases are refined, merged or split and finally grouped by the means of components. The "swim lane" representing the

7 The relations between the single use cases is represented by the use case overview in content item 3.5.2.2.1 in the system vision.

envisioned application is split into several ones representing several components. A clear differentiation between external actors and single components of the envisioned system cannot be made explicit by the established modelling techniques. Hence, the definition of a stop criterion for refinement of the use case actions respecting the methodological handling at the exclusive abstraction level of the information system hierarchy (seeing the application exclusively as a whole black box) is elaborate and remains to experienced-based best practices (see also [Coc00]).

Example 3.10. *Use Case using UML Activity Diagrams*



3.5.3.3.2 Functional Requirements

Functional requirements describe what functions the application shall offer and are often referred to as "The system shall do something"-statements. Most functional requirements of a development project are covered by the defined use cases (content item 3.5.3.3.1). The concrete interdependencies between functional requirements and the use cases consists in the level of abstraction that implicates two different concept types. While structured scenarios describe specific sequences of interaction between actors and the application as a whole (at the information system service hierarchy), a functional requirement constraints the system in single actions, consisting of an actor action and a system action as a response. Still, not all functional requirements are elaborated and covered by the description of use cases, as use cases exclusively define users' expectations from the perspective of business tasks to be supported.

Hint: *Use Cases do not cover all possible Functional Requirements*
When describing use cases, each action within a scenario of a use case can be seen as an individual functional requirement. Still, use cases do not cover all possible functional requirements, they illustrate just the overall behaviour of the application from a user's perspective in the context of the business tasks. Additional functional requirements can be a demanded communication with an external application, that that can arise from demands of IT departments.

Figure 3.22 (depicted within the content item describing the use case model) illustrates the concept used to define functional requirements.

According to [Dav93], we define functional requirements as follows:

Definition: ***Functional Requirement***

Functional requirements define the demanded external behaviour of an application that is given by a stimulus, followed by an expected response, as part of a reaction to this stimulus, independant from the underlying realisation of this response. Delimited from the term *functional requirement* are the characteristics and conditions, which are related to the expected response.

Therefore, a functional requirement is triggered by a certain *condition* that implicates an *actor* to perform an *action* while the application is supposed to *process* an *information system object* leading to an expected result. A general example would be: "When the bank employee (actor) enters the name of the customer (condition), the application shall perform a search function (action) based on the customer data (information system object)."

Finally, we delimit from functional requirements any kind of especially non-functional characteristics that are related to the functional requirements.

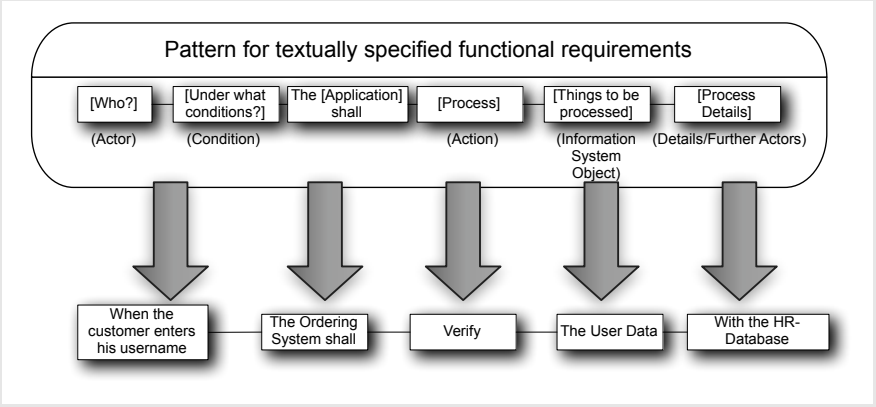
Hint: *Delimiting Functional Requirements from System Quality Requirements*
It is often said, that functional requirements build part of system quality requirements. In fact, the ISO Std. 9126 [ISO03] defines functionality as a part of the quality characteristics, in particular as part of the correctness, what in turn can be seen as the characteristic "functionality". In fact, not a functional requirement builds part of correctness, but the achievement of a functional requirements by the developed system. Hence, functional requirements and system quality requirements are different concepts as they concern different aspects [Gli05, Gli07] ("Does this requirement construct a function or does it constrain it in its properties and conditions?").

Content Dependencies. Each functional requirement relates to at least one action of a structured scenario. Any other relation to e.g. actors is given as a consequence of the interdependencies of the actions to surrounding concepts. As functional requirements are often derived from other concepts as it is the case for use cases, functional requirements can especially relate to business rules (see content item 3.4.3.1) and further informal sources like existing technical specifications or workshop protocols of discussions with IT departments.

Syntax. Functional requirements should be allocated to single use cases to which they relate (e.g. by reproducing the content item for each defined use case). Regarding their syntax, functional requirements can be represented by formal means and informal means. A formal representation can be achieved by the use of e.g. temporal logic, as referred by techniques that are based on goal-based refinement [vL03]. Although the advantages of a formal representation is understood, the target groups of business information systems mostly imply a textual (informal) representation of functional requirements in natural language. To tackle the problem of unambiguously stated functional requirements in natural language (e.g. caused by linguistic defects as described by Kof in [Kof05]), patterns can be used to support a sentence construction according the content and the dependencies defined within the concept model. The use of patterns have been shown by Fleischmann in [Fle08] and Rupp in [Rup07].

Example 3.11 illustrates a pattern that fits the single elements of a sentence in natural language according the concept model.

Example 3.11. *Functional Requirement in Natural Language*



3.5.3.3.3 Information System Service Model

The information system service model describes single pieces of functionality that are offered by an information system. Figure 3.23 illustrates the underlying concepts and the dependencies to surrounding ones.

As already described within the business service model (in content item 3.4.4.1) and in Sect. 1.5.1.2, services are often set equal to specific technologies and / or interchange formats as known from web services. In fact, information system services are not related to any kind of such technology, they are just an abstraction of use cases, i.e. a blackbox description of the possible interactions between actors and the system as a whole. The information system service model gives therefore logical representations of single pieces of functionality of an information system [TOG09], that can be called by an actor without necessarily describing any kind of interactions sequences. Although being (also) often set equal to use cases, they serve therefore as an alternative to the specification of use cases. This is for example relevant in development projects concerning hybrid systems that consist of standard as well as custom software, thereby in cases where interaction sequences and the participation of actors are not in the primary scope (see also the following chapter 4 describing the customisation approach).

However, each information system service defines the support of one or more business tasks and consequently of at least one provided business service. They describe functions of the information system that in turn, are a specific set of operations on a set of information system objects (see also content item 3.5.3.4). Furthermore, they can be decomposed to the level of abstraction of functional requirements in which they map exactly one input to one output, while the difference between both concepts is that actors remain not mandatory in the specification of the services.

According to [BKM07, Bro05, Bro03, HT09, EHH+08], we chose the following definition for information system services:

Definition: ***Information System Service***

An *Information system service* defines a piece of system's functionality to support the realisation of business processes and / or business services. It can be hierarchically decomposed and maps an input (stimulus) to an output (response).

Synonym: "Application Service", "IT Service"

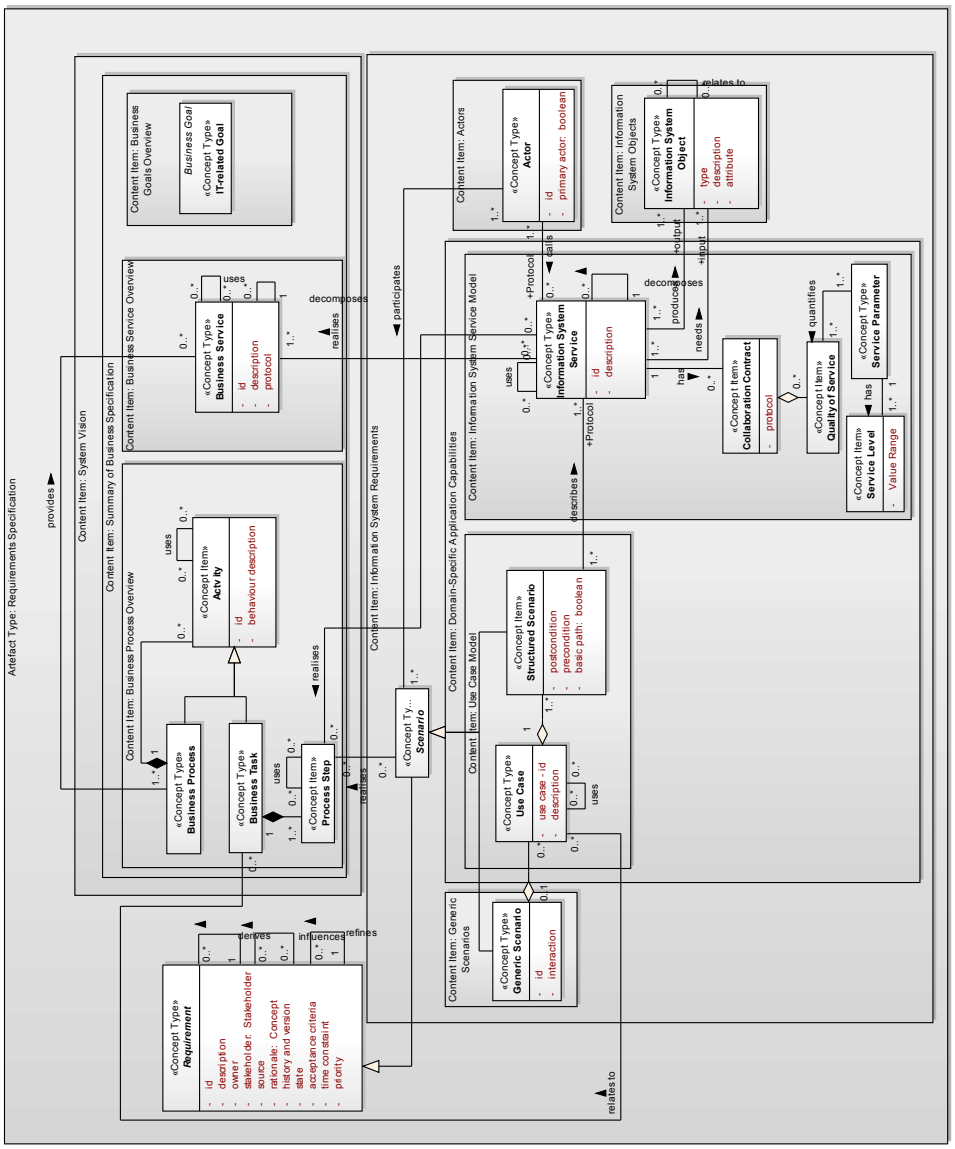


Figure 3.23.: Overview on *Information System Service Model* and underlying Concepts

Note that although each information processing system has a set of states, we do also not necessarily describe any states within the information system service model. When describing the services during initial stages of development, we do not describe the states since the system is seen as a black box and the knowledge about a reasonable set of states is weak. During further refinement of the services we can define detailed state machines at the basis of the growing knowledge about the component architecture and by making explicit design decisions.

However, For each of the described services there exists the possibility of defining collaboration contracts. A collaboration contract describes expected characteristics of an information system service in terms of how actors intent to use this service, implication also optionally a usage protocol that defines an ordered sequence of in-

teractions. For each collaboration contract there exists furthermore the possibility to define an expected quality of service. We define the quality of service according to the IT infrastructure library (ITIL) [oGCO07] and Brenner et al. [BGN06] as follows:

Definition: ***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.

Synonym: "Service Level Agreement"

The quality of service defines the expected quality when calling the information system service and possible penalties if not fulfilling the quality. Note that the measurements (the service levels) are not defined as specific measurements that have to be fulfilled at a specific point in time (e.g. measured during acceptance tests), but affect the whole service life cycle. Obviously, the service levels, thus also the penalties, are often categorised into specific ranges, e.g. silver or gold while silver defines for example the average availability of 96% and gold an average availability of 98%, measured over a specific time frame.

Finally, the concepts for the construction of the quality of single services and the ones for system quality requirements are similar. In fact, a quality of service is a subset of system quality requirements [BGN06] respecting the possibilities of expressing needs towards the system's quality. See also content item 3.5.3.5, where we explain the dependencies between both concepts.

Content Dependencies. Each business service can be supported by an information system service, while the latter then consequently supports the business processes that provide the business service. If specifying an information system service, it relates to at least one information system object that is used as an input and an output, thereby defining the behaviour in terms of such a mapping (input to output). For each information system service, there may exist several collaboration contracts that can define the order for the mapping of inputs to outputs, thus, the sequences of interactions in which actors can participate. Furthermore, as part of a collaboration contract, a quality of service can be defined that is quantified by at least one service parameter (chosen metrics), including for each at least one service level that has to be achieved when calling the service.

Syntax. Due to their nature of being contractual agreements, the information system services and underlying concepts like the quality of service are defined textually in natural language including additional diagrammatic representation using structural diagram types, e.g. given by the UML [OMG07a, OMG07b] (mapping inputs to outputs). In addition to this description, the collaboration contracts, in particular the protocols, can be defined via input/output tables as described by Hummel et al. in [HT09]. The advantage of such a notion is that the behaviour can be expressed without the explicit definition of actors, sequences and in particular system states that are all not in scope of the concept.

3.5.3.4. Information System Objects

While information objects define what is carried out by single user groups while performing their business tasks, the information system objects define what specific subset of information objects is directly related to and processed by the envisioned application. Hence, they are a system-side representation of selected information objects that are processed by the application when calling an information system service and / or executing a use case. Figure 3.24 illustrates the concept on its right side, being surrounded by the information objects within the vision and the domain-specific application capabilities that make use of the information system objects.

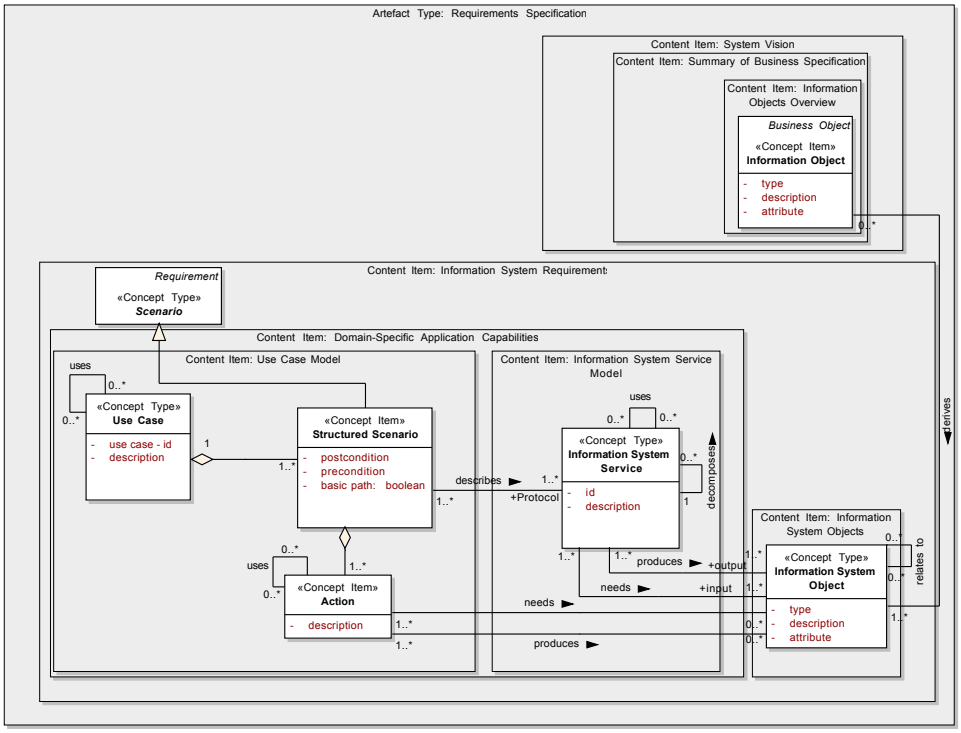


Figure 3.24.: Overview on *Information System Objects* and underlying Concepts

This usage is in particular performed in terms of *creating*, *reading*, or *updating* the information system objects. For example the operation "create" (new order) that is related to the structured scenario "place order" being performed by a waiter in a restaurant. When surrounding information system services and structured scenarios make use of information system objects, they implicate (directed) relations with cardinalities between the objects. For instance, if the scenario "account order" producing several instances of the information system object "Bill" needs the output object "Order" of the fore going scenario "place order", this implicates a directed relation from the object "Order" to "Bill", such as each order having at least one associated bill.

However, each of the information system objects encompass a description of the:

- Type: The name of the information system object is its unique identifier and represents a superset of the information objects from which it is derived. For example the type "Order".

- Description: The description of the information system object provides a quick overview of the context and its content. For example "The order that is taken within the restaurant and processed by the point-of-safe system"
- Attribute(s): Attributes describe the content that is encompassed by the information system object. For example the attributes "Meal" (that is ordered) and "Desk" (that ordered the meal).

Finally, we define the concept as follows:

Definition: ***Information System Objects***

An *information system object* describes the system-side representation of an information object that can be processed in terms of being created, read of updated.

Content Dependencies. Information system objects are referred by structured scenarios and / or information system services. The relation to those concepts is described within the concepts that make use of the information system objects. The relation to the information objects from which they are derived is given by the type of the information system objects.

Furthermore, information system objects can relate to each other, while this relation is given as a consequence of the relations in-between the actions of single structured scenarios and / or the mapping of the input to the output within the definition of information system services, both implying an ordered sequence of operation on the objects causing directed associations between the objects.

If and how to document the relations, depends on the knowledge on the system's internals, e.g. the communication between single components. Hence, even if documenting relations, these still might change during solution crafting.

Hint: *Information System Objects versus the Logical Data Model.*
The difference between the information system objects and the logical data model is given by the relations between the objects. When the relations are given and defined, single objects are set into relation as part of a model.

Syntax. How to document the information system objects depends on the complexity of the application's logic, i.e. how many operations are performed on information system objects. Consequently the objects can be defined as part of a list that summarises all the processed objects or as a model that emphasises the relations using structural diagram types like UML class diagrams [OMG07a, OMG07b].

3.5.3.5. System Quality Requirements

System quality requirements may decide on the success and failure of projects as the main challenge consists in developing the right application in an appropriate customer-adequate quality. At the same time, they build an aspect that is often neglected, because they are difficult to elicit, to refine and to implement. In fact, there still exists no established definition of system quality requirements, caused by different understanding on quality [Gli07, KP96].

The most important and at the same time difficult aspect regarding the handling of system quality requirements consists in (1) capturing the requirements precisely avoiding unambiguous not measurable statements, while (2) not directly constraining the decision on how to realise the requirement (leaving degrees of freedom for architects), (3) but still describing exactly the requirements in a sense that is driven by the needs of customers.

In the context of this report, however, we refer to system quality requirements as described in Sect. 3.2.3.2 as "How good must activities be supported by systems in order to achieve defined business goals?" The term *system quality requirement* is chosen according to the ISO Std. 25030 for quality requirements [Boe08] and exclusively concerns aspects of a system in order to achieve some outcome of value.

Hint: *Term "Non-functional requirements"*
Non-functional requirements specify everything that is not functional, e.g. until when a specific release has to be delivered or if developers must undergo a certification. Everything inside a requirements specification — except functional requirements — describes therefore non-functional aspects of the whole project. In this report, we consider system-specific quality requirements and therefore delimit from the term non-functional requirements.

However, as stated in Sect. 3.2.3.2 we perform the refinement of system quality requirements by a continuous decomposition of business goals that steer the description of behavioural aspects and characteristics of the business and the underlying systems [WMFIL09, WDW08, DJLW09]. The behaviour describes business activities as well as behaviour from a user's perspective when using the system.

Figure 3.25 illustrates therefore in its bottom part the concepts of the system quality requirements and their interrelation to concepts used to describe (besides other aspects) the behaviour, situated in upper parts of the figure.

System quality requirements are initially captured as part of IT-related goals like maintainability or availability. They describe high-level characteristics of a system, named quality attributes (see also content item 3.4.3.2).

These IT-related goals are used to establish appropriate generic scenarios that describe user-visible characteristics and behaviour of the system by means of sequences of interaction with the system and its environment (see content item 3.5.3.2). For instance, what modifications an administrates does with the system during run-time when performing his maintenance tasks and how he expects the system to behave in order to achieve the maintainability.

Even if generic scenarios can often be easily tested, there exist an arbitrary number of possibilities of implementing this scenario with an arbitrary number of cost-intensive design-decisions that perhaps do not meet the expectations of the customer [CNYM99]: "What means during run-time and as how large may this modification be considered? Are users operating on the system? How many? ...". Anyway, generic scenarios often build the only possibility of expressing system quality requirements and at least remain at a better glance as writing "the system shall be maintainable".

Still, system quality requirements are to be seen as a quantification of generic scenarios, in which the demands are stated with proper measurements that make the requirement measurable or at least assessable, for e.g. software controlling or testers.

The system quality requirements then constrain specific properties that behavioural or structural parts of the system must exhibit in order to enable the generic scenarios and consequently support the business tasks from which the generic scenarios have been derived. These properties take influence on one or more quality attributes, because they are related to specific characteristics that the customers perceive.

Hint: *System Quality Requirements constrain properties of the System and not Quality Attributes*
System quality requirements are often meant to constrain quality attributes of a system [DKVKP03]. Indeed, system quality requirements affect quality attributes (e.g. increasing the usability) but do not directly constrain them. They constrain the system

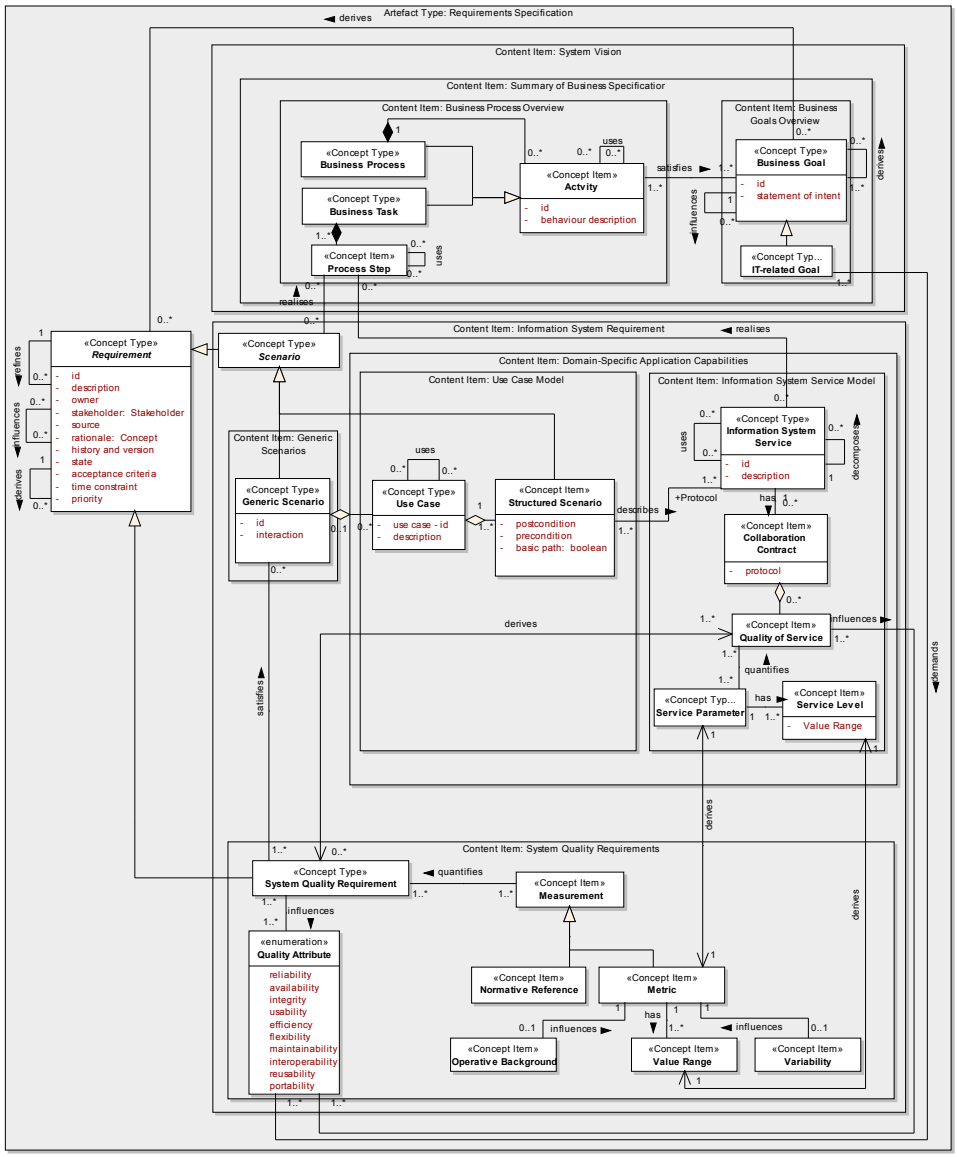


Figure 3.25.: Overview on *System Quality Requirements* and underlying Concepts

and its environment in their properties e.g. the language or the simple existence of the documentation. A documentation takes effect on the maintainability, but also on the usability of the system.

However, another possibility for defining system quality requirements is given by defining the quality of information system services. It serves as an input to defining system quality requirements (and vice-versa), as subsequently described within the content dependencies.

Corresponding to these views, we define system quality requirements in the context of this report as follows:

Definition: *System Quality Requirements*

System quality requirements constrain a system and its (development and usage) en-

vironment in order to support an activity, thus in order to satisfy a business 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.

Examples:
(1) Demanded response-time of a search function
(2) Demanded completeness of the documentation

To sum it up, we subsequently describe the content when specifying system quality requirements according to the concept model:

- *Affected quality attribute*: Although system quality requirements affect quality attributes only indirectly, the requirements should be organised by the like in order to support the communication to customers.
- *Constrained elements (of the system and its environment)*: The requirements can either constrain general elements (e.g. the documentation in general) or specific elements (e.g. "the user interface 'Configuration-Mask' ")[8]. This depends on the actual knowledge on the system's realisation and the actual degree of decomposition. System quality requirements in general constrain behavioural elements or structural elements. The latter comprehends the system itself (e.g. user interfaces, interfaces to external applications, components, ...) and its environment (e.g. the documentation). When initially defining system quality requirements we often constrain specific functions e.g. in their response-time. As these functions are often described by means like use cases, this often this leads to specifying system quality requirements inside these use cases, while we take a critical stance towards this approach. The reasons are that (1) this only works well for some mostly performance-related cases, (2) we also would have to state the requirements in other affected elements what would cause redundancy and finally (3) system quality requirements can affect due to their cross-cutting nature several elements of the same type, like every function that accesses a certain database.
- *Metric*: A metric reflects a non-ambiguous measurement for a test that clearly can be answered with "Yes, we are compliant" or "No, we failed to realise the requirement". Metrics ensure that following constructive and surrounding analytical phases are able to unambiguously interpret, realise and control, respectively test the compliance of the system to the requirement. For example, "response-time in seconds", "mean-time to failure (MTTF)", "number of jobs per minutes", "existence", etc.
- *Value range*: According to a chosen metric, a corresponding value range must be specified. An example would be defining "3-5" for the metric "response-time in seconds".
- *Operative background*: It is often necessary to document expected operative backgrounds. Examples for expected operative backgrounds include the expected transaction loads, storage loads or the number of expected parallel executions of certain functions. The operative background also includes technical assumptions involving the existing underlying network infrastructure and its realisation.

8 Usually, system quality requirements affecting more the external quality like efficiency constrain specific elements, requirements affecting more the internal quality like maintainability constrain more general elements.

- *Normative reference*: Normative references reflect guidelines describing properties of structural elements. They help in describing the requirements at an assessable level, especially if not being able to define a corresponding metric. An example for such a normative reference could be a norm for web accessibility or a user interface design guideline.

Content Dependencies. System quality requirements can be derived from generic scenarios and influence at least one quality attribute. Their statements must be described by at least one metric or normative reference, while each metric has at least one assigned value range. Furthermore, in relation to a metric we can define one variability and / or one operative background description.

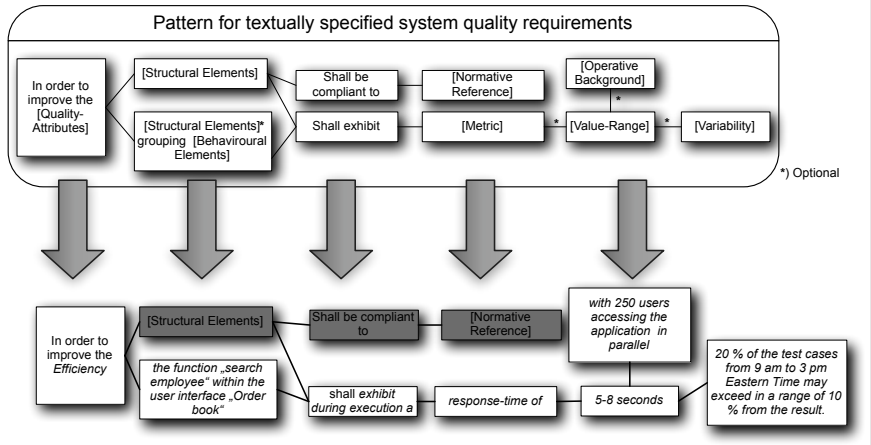
A final dependency is given to the concept of information system services, in particular to the quality of service (see content item 3.5.3.3.3). The quality of service builds regarding the possibilities of stating quality needs a subset of the system quality requirements. Both concepts have the possibility of expressing needs on external quality characteristics and characteristics of quality in use in common [ISO03], i.e. user-visible characteristics that are observable when calling a service. Aspects concerning more the internal quality of the system, such as constraining the system's environment for the purpose of increasing maintainability-related characteristics are not in scope when formulating the quality of a service. However, if overtaking system quality requirements from the quality of service (or vice-versa), the service parameter corresponds to the metric, while the service levels correspond to the value range. Further going information can be taken from [BGN06].

Syntax. There exist two ways of representing system quality requirements:

1. by the use of rich components.
2. textually written in (structured) natural language.

The rich component model, developed by Damm et al. [Dam05, DVM+05], arises from the application domain of embedded systems and offers means to describe functional and system quality requirements as part of a component model. It enriches the concept of components by e.g. viewpoints, promises and assumptions. Still, the approach is not adaptable to the needs of the model at hand.

Example 3.12. *System Quality Requirement in Natural Language*



The reason is that we do not describe at this stage of development any kind of components. In addition, although being a promising approach it is not evaluated to what extent the approach supports the definition of quality aspects that address the internal quality (like maintainability). Hence, we refer to system quality requirements that are written in natural language.

Likewise to the syntax of functional requirements in content item 3.5.3.3.2 we give with example 3.12 guidance with a pattern for a sentence construction that is conformant to the concept model. The asterisks within the pattern symbolise optional paths and elements.

3.5.3.6. Architectural Constraints

Architectural constraints restrict the solution's current and future architecture orthogonal to functional requirements and quality requirements without giving any degree of freedom for realising the constraints. They restrict the architecture of an information system in its logical or technical layer and (partially) emphasise the "How" of a system, rather then demanding "What" shall be done in general. This is done either by directly demanding specific solutions, or by excluding specific solutions. For example by demanding the compliance to architectural standards, such as SOA-conformity, by defining restrictions on interfaces and communication protocols to surrounding external systems, or by demanding the use of customer-side existing technological platforms.

The architectural layers that are in scope of architectural constraints are:

1. the logical layer that describes how the business logic is structured and reflected from an architectural perspective ("How will the system be structured?"), and
2. physical layer ("With what will the structure be implemented?").

The latter includes aspects of hardware and used technologies that supports the realisation of a system like an oracle databases or to a cisco network router. Figure 3.26 illustrates the concepts of architectural requirements and the dependencies to surrounding ones.

We define the concept as follows:

Definition: ***Architectural Constraint***

> *Architectural constraints* are statements that restrict the information system's solution in its architecture respecting its logical and / or physical layers. They demand or exclude specific solutions.

However, we distinguish because the different architectural layers that are in scope of the architectural constraints between two different concepts:

1. logical restrictions
2. technical restrictions

3.5.3.6.1 Logical Restrictions

Restrictions on the logical architecture define how to structure the architecture in terms of how to arrange the components and their (communication) relations affecting the definition of the layers of the architecture. We define the restrictions on the logical architecture as follows:

Definition: ***Restrictions on the Logical Architecture***

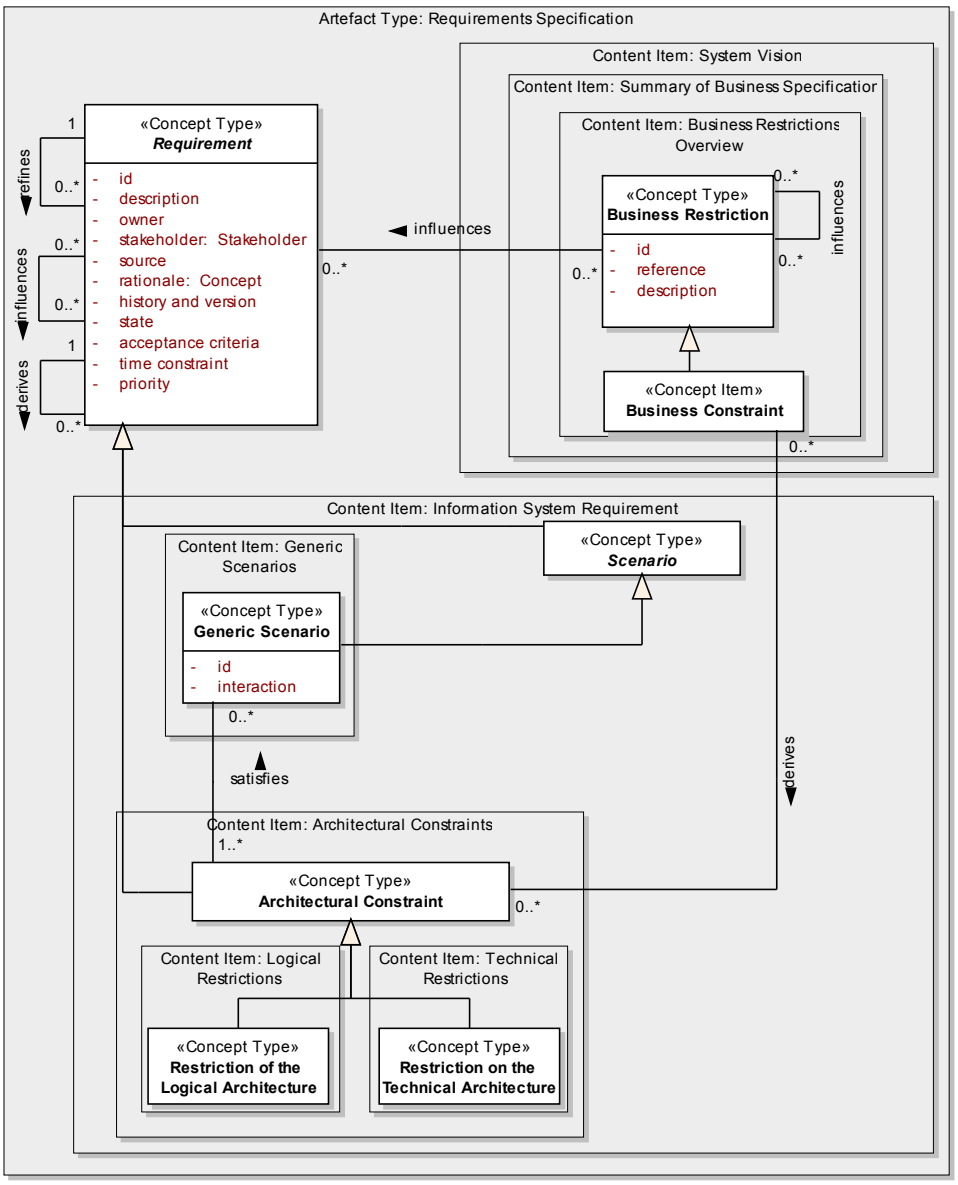


Figure 3.26.: Overview on *Architectural Constraints* and underlying Concepts

Restrictions on the logical architecture restrict the of the solution's architecture logically in its structure.

The restrictions on the structure can be indirectly stated as a demand towards the application that results from the needs of the business logic (affecting the architecture) or can be directly stated as a restriction on the structure of the architecture.

An example for the first could be "The application shall enable an outsourcing of the share management processes", thereby restricts the structure of the components to encompass all functions that enable the share management (business) processes. An example for the second could be "The application shall be realised as a rich client" or "The architecture must be compatible to a 3-tier architecture".

Content Dependencies. Each restriction on the logical architecture can be derived from one or more business constraint (see content item 3.4.3.1) and can be expressed by one or more generic scenarios (see content item 3.5.3.2).

Syntax. Because of the informally defined sources from which the architectural constraints are derived and because of the (customer-side) target groups, architectural constraints are specified textually in natural language. Furthermore, as the constraints can strongly vary in their assertions there is no possibility of standardising their content, thereby define patterns for a sentence construction as done within functional requirements.

3.5.3.6.2 Technical Restrictions

Restrictions on the technical architecture define what technologies to use for the implementation of the logical architecture and restrict also the hardware of the architecture. They result often as a consequence of the existing technical infrastructure into which the system under consideration has to be integrated and / or are enforced by existing licenses to which customers are restricted. Examples for such restrictions could be "Information system services shall be realised using Web Services", "The connectivity to the database must be realised using the Oracle Transparent Gateway", "Object-relational mappings must be realised using Hibernate", or the simple exclusion of open-source technologies.

Definition: ***Restrictions on the Technical Architecture***

Restrictions on the technical architecture restrict the architecture of the solution in the used technologies and used hardware.

Content Dependencies. See the dependencies of the logical restrictions in content item 3.5.3.6.1.

Syntax. See the syntax of the logical restrictions in content item 3.5.3.6.1.

3.5.4. Integrational Requirements

Integrational requirements are requirements that restrict the execution of the integration and the delivery of the system in its process and in its technical details. They serve as a basis for planning a release strategy. Figure 3.27 illustrates two concepts and related content items that are used to define integrational requirements:

1. deployment requirements restricting the integration process in organisational aspects and the technical environment during initial launch, such as "During re-launch of the application training courses shall be offered to the employees"
2. migration requirements describing restrictions on the replacement or modification of one or more legacy applications, such as "The list of the data elements to be transferred is as follows: [name], [type], [alternative format],..."

3.5.4.1. Deployment Requirements

Deployment requirements emphasise (1) organisational aspects such as training courses, structure and order of the deployment and (2) technical characteristics

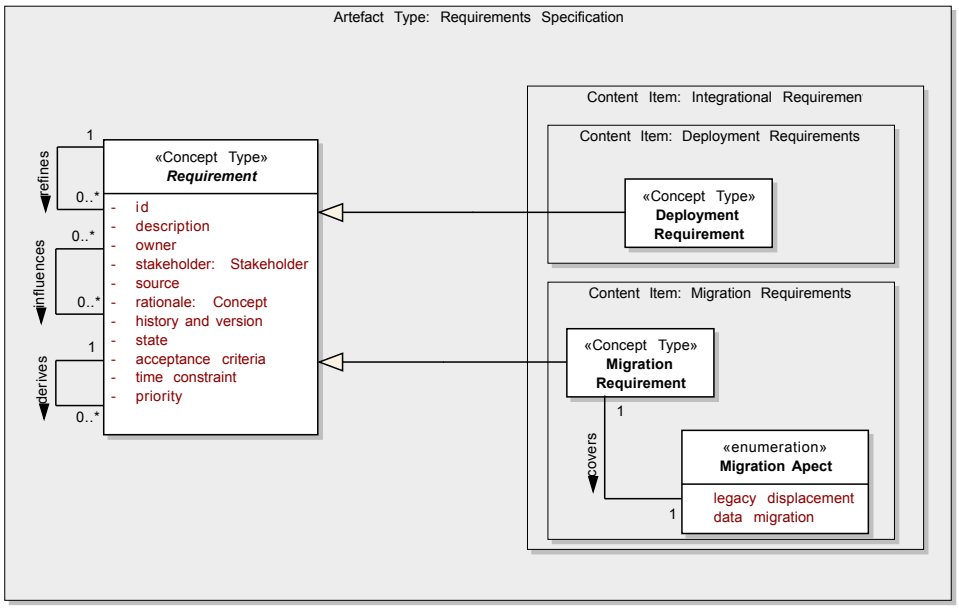


Figure 3.27.: Overview on *Integrational Requirements* and underlying Concepts

that go beyond the migration of legacy systems and operating issues, such as technical environments that are used during the deployment (see content item 3.5.4.2).

Definition: ***Deployment Requirements***

Deployment requirements describe 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.

Content Dependencies. None, except the general dependencies between the requirements elements themselves.

Syntax. As deployment requirements can vary in their concern affecting organisational and technical aspects, they are not standardised, thereby are textually listed.

3.5.4.2. Migration Requirements

Migration requirements describe demands towards one of the following two aspects:

1. the displacement of legacy systems emphasising needs towards the strategy (e.g. respecting sequential steps for displacing legacy systems).
2. data migration from legacy systems to the envisioned one, covering the data elements to be transferred and used mapping rules to the information system objects of the envisioned system.

Aspects of used technologies are not in scope of migration requirements.

Definition: ***Migration Requirements***

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.

Content Dependencies. The concept exhibits no explicit dependencies. Still, note that migration requirements concerning aspects of data migration can only be completed when the definition of the overall information system objects (also from the given external systems) is completed.

Syntax. The syntax for representing migration requirements depends on the covered migration aspect. If concerning the displacement of legacy systems, the steps can be represented with charts, or project plans (emphasising ordered steps to be performed rather than specific time schedules). If concerning, however, the data migration, we use tables for describing the mapping of the "old" data model to the new one. If data conversion algorithms need to be described, this can be performed by the use of pseudo code.

3.5.5. Organisational Requirements

Organisational requirements, often described as process requirements, restrict the overall project execution and its process.

Definition: ***Organisational Requirement***

Organisational requirements define the demands towards the project execution and its process in general, including project requirements and obligations.

Organisational requirements can either be:

1. obligations concerning additional agreements, propositions and exclusions between selected parties ("What shall be done by whom and until when?"), such as "The development process and the corresponding working products shall be compliant to the German standard V-Modell XT"
2. project requirements affecting the process model ("How shall it be done?") and the artefact types, for example by defining standards to which the artefact structure has to be adapted, such as "The Media API will not be supported. The specification of the interface to the external application 'SAP HR' will be delivered by the customer within the release [release]"

Figure 3.28 illustrates the envisioned concept.

3.5.5.1. Obligations

We define obligations as follows:

Definition: ***Obligations***

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.

Content Dependencies. None, except the general dependencies between the requirements elements themselves.

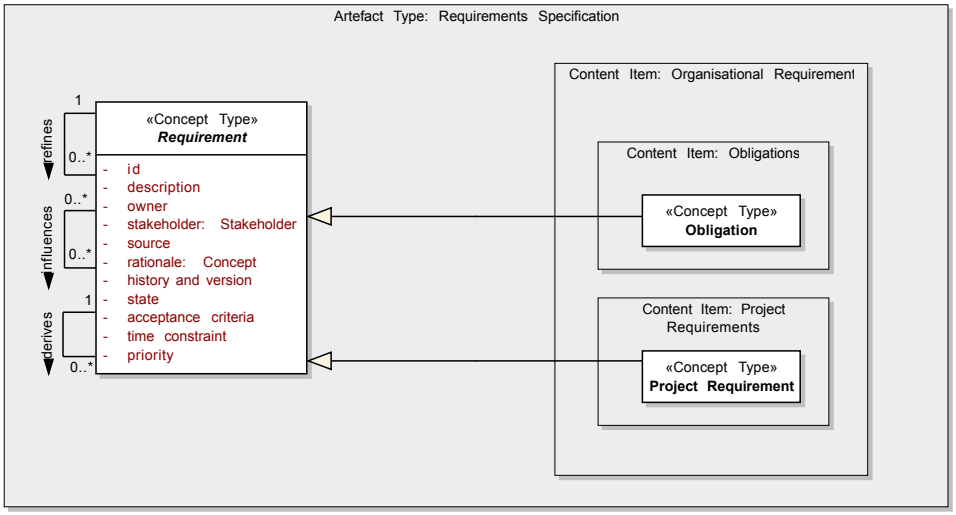


Figure 3.28.: Overview on *Organisational Requirements* and underlying Concepts

Syntax. Obligations are described textually in natural language.

3.5.5.2. Project Requirements

We define project requirements as follows:

Definition: ***Project Requirements***

Project requirements are constraints towards the content and / or structure of selected artefract types and the process model, i.e. affecting the process design.

Project requirements affect the process as well as the artefacts. The process can be affected in its milestones regarding time schedules, used infrastructure like mandatory tools, and compliance to selected standards and approaches like to the V-Modell XT. The artefacts can be affected in their structure by e.g. enforcing the structure according to selected standards, such as the *IEEE Std. 830-1998 recommended practice for software requirements specifications* [oEEE98].

Content Dependencies. None, except the general dependencies between the requirements elements themselves.

Syntax. Regarding the syntax, project requirements can be described textually in natural language using in addition diagrammatic representations for the description of time schedules, like gantt charts.

3.5.6. Requirements Risk Status Report

The requirements risk status report summarises all the risks that arise from single defected requirements. It describes the requirements that exhibit certain errors and defects and serves as an input for risk management activities (see also Sect. 3.8). A requirements error is a mistake that has been done during project execution, a misconception of the actual needs of a customer. A requirements risk is a

consequence of certain defects within the requirement — a manifestation of an risk factor. For example, a requirement could state "The system shall provide sufficient access control [...]". The requirement would exhibit the defect of being ambiguous ("What means sufficient?"). The risk factor could be the lack of technical knowledge of the stakeholders.

Definition: *Requirements Risk* and *Risk Factors*

A *requirements risk* is a defected requirement that causes potential problems to the development activities and to the final software product quality due to specific risk factors.

A *risk factor* is the root cause for the requirements risk. One risk factor can be the root cause for several requirements risks (defected requirements).

However, the requirements risk status report aims at controlling risks that are related to requirements and structures the risks according to a specific rating (the risk score), also describing how these risks can and should be mitigated. Figure 3.29 illustrates the used concepts.

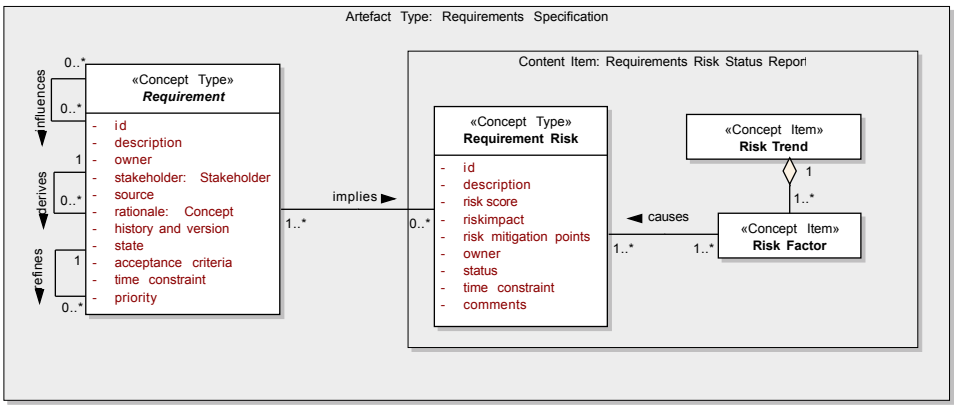


Figure 3.29.: Overview on *Requirements Risk Status Report* and underlying Concepts

Each requirement that is defected can be assessed by specific test criteria and assessment strategies. One possibility of performing this assessment is given by defining test criteria for chosen (possible) defects, e.g. by defining questions that address if requirements are unfeasible, incomplete or incorrect. In dependency to the answers, the risks are detected and rated. The result of this assessment is the risk score and possible mitigation points. The risk score gives us the possibility to rate the risks, while the mitigation points are used to control the risks. The rating can be taken for adjusting the priority of requirements (see also the defined requirements attributes).

As a whole, we define the following content within the risk status report for each of the identified faulty requirement:

- *ID*: The ID of the defected requirement is captured.
- *Description*: A short description of the risk details gives a quick overview on the defects of the requirements based on the risk factors (such as "unfeasible").
- *Risk score*: The risk score determines the value of the risk with specific ranges. How exactly the score is calculated depends on the chosen metrics.

- *Risk impact*: Depending on the risk score a specific risk impact identifies the severity of the risk. One possibility for defining the impact levels is given as follows: catastrophic, high, medium, low and none.
- *Risk mitigation points*: Mitigation points specify the appropriate treatment actions that are meant to control the risks.
- *Owner*: The owner is the person who takes the responsibility mitigate the risks.
- *Status*: The status of a requirements risk provides the update information on its actual degree of mitigation.
- *Time constraint*: The time constraint specifies the estimated time until when the risk needs to be solved.
- *Comments*: Additional comments can be made, if required.

In addition to the risks that are directly related to requirements, there exists the possibility of defining risk trends. The main idea of a risk trend is to identify and summarise the defected requirements according to their frequency of appearance. This facilitates to focus more on the root causes so that same and similar risk defects might not further arise. A risk trend therefore quantifies the requirement risks to a summary and shows how often same and similar risks come up within the same project. It can be expressed by a percentage that is determined by counting how many requirements are affected by the same defects.

It also counts the influential risk factors themselves, what leads to a risk factor trend.

Definition: *Risk Trend* and *Risk Factor Trend*

A *risk trend* calculates the total amount of requirements that are affected due to the same defects.

A *risk factor trend* calculates the total amount of risk factors that are the root cause for the defects.

Example 3.13. *Exemplary Risk Trends and Risk Factors Trends*

Requirements Risk Trend:

- *40 % of the defected requirements are incomplete*
- *20 % of the defected requirements are unstable*
- *...*

Requirements Risk Factor Trend:

- *30 % of the requirements risks are caused by a missing technical knowledge of the customer*
- *15 % of the requirements risks are caused by a missing user involvement*
- *...*

Content Dependencies. Each requirement can imply one or more requirements risks that are caused by at least one risk factor. The risk factors build part of one risk trend.

Syntax. The risk status report should be defined textually within a spreadsheet. This also depends on used risk management standards (see also Sect. 3.8), so that the risk status report should be adapted to the like.

3.5.7. Glossary

The glossary within the requirements specification extends the one of the business specification with all the requirements-specific, mostly technical, terms that have influence on the requirements specification (e.g. arising from architectural requirements). As the glossary is from the perspective of the used concept same to the one in the business specification, it also has no dependencies.

Syntax. We define the same syntax as done within the business specification (see also content item 3.4.8).

3.6. Artefact Type: Traceability Matrix

Traceability in general concerns "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 specication, 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 and furthermore ensures that each developed component of the system satisfies a requirement [Ste90, Wri91]. The term traceability is defined according to Ramesh et al. [RJ01] as follows:

Definition: ***Traceability***

Traceability is a characteristic of a system and its development in which the requirements are clearly and unambiguously linked (1) backward to their sources and rationales and (2) forward to the artefacts that are produced during following constructive disciplines based on the requirements. The documented links 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.

The traceability matrix, however, documents all the requirements traces, maintaining links between the different artefacts in order to follow the evolution of a requirement throughout its life cycle.

Definition: ***Traceability Matrix***

The *traceability matrix* describes the requirements traces that are produced over the whole development life cycle, defining the backward traces and the forward traces of the requirements artefacts.

The traceability matrix is initiated during business information systems' analysis, but extended during subsequent development phases and production of further artefact types. Obviously, the matrix builds an own artefact type as its life cycle accompanies all development activities.

As the traceability matrix captures all the links between the artefacts of different software development phases, it is an effort to ensure the consistency in-between these links. In fact, the advantage lies especially in large projects as the traceability matrix enables:

1. Impact analysis during a change management procedure, describing what elements of constructive phases are affected by modified or new requirements.
2. Controlling of the requirement by proving the degree of appliance of the overall amount of requirements, i.e. describing the functional coverage of system use cases ("How many requirements are realised by this system use case?")
3. Reusing requirements, enabling the proof and comparison of traceability matrices across different projects (e.g. "What did others do to refine and realise this quality requirements and how much did it approximately cost")
4. Detecting conflicted requirements ("Conflict Management") by capturing what elements of the constructive phases are affected by same and similar requirements that possibly make contrary assertions
5. Checking the rationale of requirements and discovering necessities of readjusting the business specification (e.g. "are there business goals or business processes that are not satisfied by any requirements?")
6. Finally, discovering the history of implemented function and design decision to ward of gold-plating [Pal91, Ste90, Wri91]

The concept of a traceability matrix represents all possible associations between the requirements classes within the requirements specification and (backward) from the requirements to the concepts of the business specification (from which requirements are derived).

Content Dependencies. The traceability matrix represents itself all the associations between selected concept of the artefact model.

Syntax. How to define and to organise the matrix depends on the used infrastructure, e.g. on if the matrix is exclusively documented within spreadsheets or managed by tools. If referring to tools the traces are managed by the stored associations between the concepts. The way of how to organise the traceability matrix strongly depends on the purpose of the matrix. In general, there exists:

1. the requirements-centred structure and
2. the artefact-centred structure.

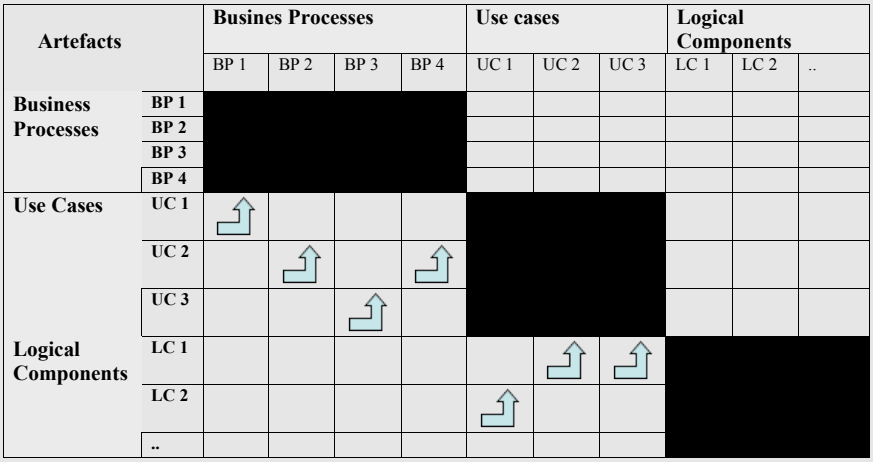
The requirements-centred structure puts the requirements in the centre of our attention. It defines for each of the documented requirements (concepts) the elements that finally realise the requirements during following design activities and that rationale the requirements during foregoing activities concerning the business specification. Example 3.14 illustrates resulting structure of the traceability matrix, often referred for the purpose of using the matrix for acceptance test cases.

Example 3.14. *Requirements-centred Structure*

Requirement	**Business Processes**				**Logical Component**			**...**	
	BP 1	**BP 2**	**BP 3**	**..**	**LC 1**	**LC 2**	**..**		
Requirement 1	X				X				
Requirement 2		X	X			X			
Requirement 3				X			X		
....									

The artefact-centred structure does not make any difference between requirements and other artefacts and structures the matrix according to any possible traces that are established between the artefacts in general. Example 3.15 illustrates the matrix, while we depict only three artefacts for reasons of complexity: business processes, use cases and logical components.

Example 3.15. *Artefact-centred Structure*



3.7. Interrelation of Requirements Concepts with System Concepts

We subsequently illustrate exemplary concepts of the requirements specification and show how they interrelate with the concepts of a system (specification). Obviously, this also depends on the chosen (system) architecture framework and the underlying concepts.

Because the system concepts are not in scope of this report, Fig. 3.30 illustrates in its centre a generic description of a system concept. It can be extended or replaced by any architecture model (see also Sect. 1.5.1.2).

In any case, each system specification is constructed by making use of three (abstract) concepts:

1. *Environment*: The environment represents all aspects that are not directly related to the system's functionality or structure. Still, it is of interest during development and integration activities. For instance, the documentation, the deployment environment, the code or the underlying network infrastructure.
2. *Structure*: The structure of a system represents the system's architecture and results from the (functional) decomposition of the architecture into subsystems and finally components. The latter offer ports that define how components communicate with each other and thereby represent the interfaces of single components. We make no distinction between components and user interfaces or further concepts that are used to describe the communication to the system's environment.
3. *Functions* (Behaviour): The functions describe behaviour that is logically structured by the components, in particular by provided applications that consist of a set of functions.

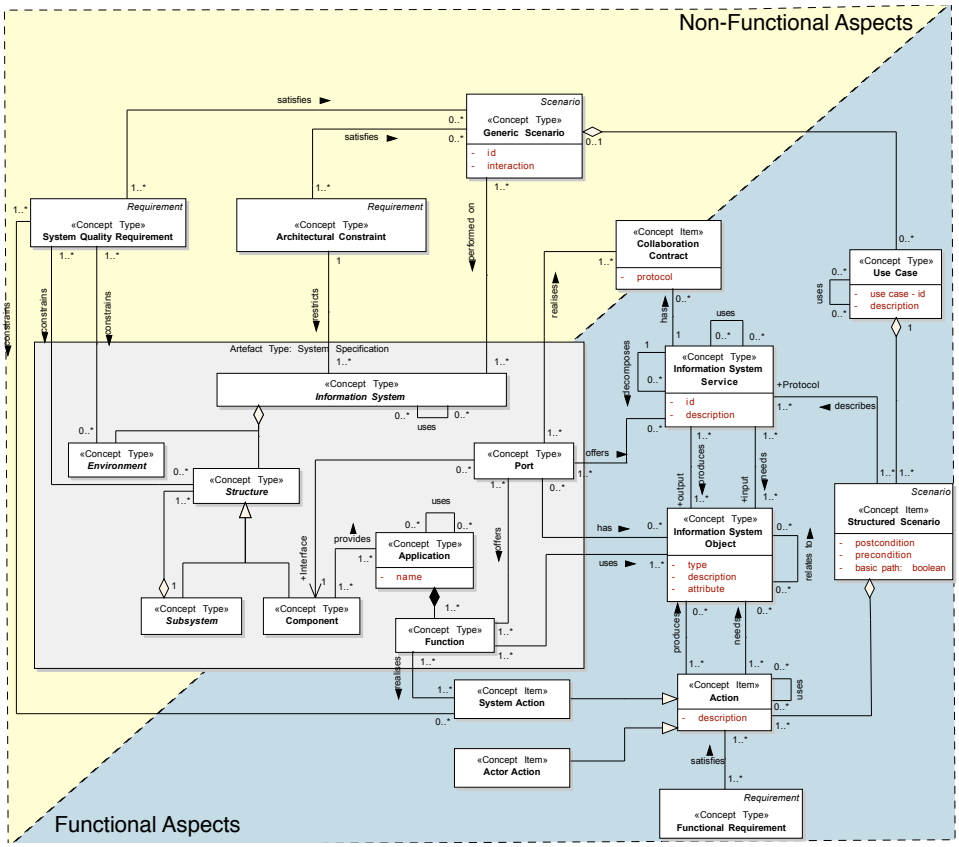


Figure 3.30.: Interrelation of Requirements Concepts with System Concepts

Regarding the concepts of the requirements specification and the transition to the solution, we distinguish between functional aspects that steer (motivate) the construction of the architecture and non-functional aspects that constrain how the architecture is constructed. Functional aspects are reflected by the elements on the right part of Fig. 3.30, non-functional aspects by the upper part.

Functional Aspects. Functionality is defined during the construction of the system's architecture by the explicit definition of functions and components being compliant to e.g. functional requirements. Functions are offered by ports as a function can be executed by "calling the port". A function is the realisation of at least one system action that is described (demanded) within structured scenarios. For instance, the realisation of the system action "search employee", offered via a port of the component "Employee Administration". Similarly, the ports of single components describe what information system services are offered, respectively what information system services are reflected by single components. In particular, the ports define how to communicate with a component and thereby realises the protocol of an information system service. As a consequence of this communication, the ports also describe what information system objects are processed during the execution of a function. They take the "functional sovereignty" over at least one object that is in relation to an information system service and to actions of a structured scenario[9]. How the exact dependencies between the services and

9 Note that actions must not refer to an information system object, while functions must. The reason is that the actions also describe actor actions that do not necessarily relate to an object, for example if

the components are, strongly depends on the granularity of the components and the one of the services. In the ideal world one (logical) component offers one or more information system service, because one component would correspond to one business domain that logically groups the services. For reasons of complexity, it is not depicted in the figure.

Non-Functional Aspects. Regarding non-functional aspects, the interrelations to the system's concepts are as follows. Generic scenarios, reflecting high-level system quality requirements and architectural constraints, define besides the relations between the use cases, any further kind of interaction with the system that is not functionally motivated by the business process landscape of an organisation. These interactions involve the system as a whole including its environment (e.g. the code). Referring to system quality requirements, they explicitly constrain one of the three concepts. The can directly constrain the environment like demanding properties of the documentation, the structure like demanding properties of user interfaces, or the functions like constraining them in their response time. Instead, architectural constraints exclusively restrict the logical architecture in its structure or the technical one, for example by enforcing the use of specific technologies.

3.8. Artefacts as Interfaces to the Development Life Cycle

The artefact model exhibits dependencies to the development life cycle respecting selected target groups that make use of single concepts. We give an overview on the interfaces to surrounding development phases in terms of depicting what content items serve as a basis for specific activities. We build therefore a bridge from BISA activities into the surrounding development life cycle. As the content of the defined artefact model has to be customised and integrated into specific process models, we describe in here only general dependencies to selected activities. How, respectively to what activities, the content items exactly relate strongly depends on the organisational environment, e.g. the chosen process model. See also the customisation approach in chapter 4).

We divide the development life cycle into three areas including generic activities that refer to the contents of the presented artefact types as an input. We refer in particular to the areas "planning & controlling activities", "design activities" and "quality assurance activities ".

Planning & Controlling Activities. Planning and Controlling activities include management aspects in the broader sense such as:

- **Bidding and Acquisition:** During acquisition and bidding procedures the *business vision* and the *system vision* serve as a basis for negotiating the main objectives while abstracting from technical details. Both are means for making trade-offs and for steering the relevant stakeholders within (potential) projects into a common direction.
- **Contract Management:** The definition of contractual agreements are also performed at the basis of the *business vision* and the *system vision* as both visions summarise the elementary scope. Regarding system development projects, the *requirements* and *features* are within the system vision of major interest. Especially features can be used as a basis for the definition of frame contracts, because they define an abstract view onto an amount of detailed

an actor request to cancel an ordering procedure.

requirements. Each feature within the contracts can then for example be coupled to specific failure classifications and corresponding penalty agreements.

- **Complexity and Costs Estimations:** The *system vision*, again, takes also an important role when performing complexity and costs estimations (regarding development[10]). It summarises in particular all the use cases and generic scenarios that are specified in detail within the *information system requirements*. This content builds the basis for performing analyses of (functional) complexity if e.g. referring to the function points analyses and further sizing analyses. This supports the estimation of the costs of a development project.
- **Project Planning:** When elaborating project plans, including the definition of team structures, work packages, milestones and time schedules, there are three content items of interest: the *business vision*, in particular the *project scope*, *organisational requirements* and *business domains*. The latter support the approval activities, e.g. in planning and performing workshops according to single business domains and related stakeholders.
- **Risk Management:** Risk management activities define how to treat any risk factors of a project. We distinguish between two major areas of risks: Risks that affect a customer's organisation (e.g. a market rejection of a business service) and risks that affect development (e.g. technical unfeasibility caused by unrealistic system quality requirements). The first is in scope of the *business demands analysis*, the latter in scope of the *requirements risk status report* while both serve as an input for general risk management activities.
- **Problem Management:** Problem management covers a broad spectrum of activities. In particular, we refer to the identification of moving targets (e.g. due to frequently changing requirements) and to the identification of approval creeps. The detection, but also the avoidance of the like, are supported by the *system vision* and by the *traceability matrix*. The latter supports for example controlling the degree of appliance of requirements in terms of what requirements remain untreated.
- **Claim and Change Management:** The identification of failures and responsibilities and the execution eventually resulting requests for a change is supported by the *traceability matrix*. It supports impact analyses in terms of indicating what parts of development artefacts are affected by changing or new requirements.
- **Release Planning and Deployment:** The release and deployment strategy of a project must be compliant to any restriction of customers that in turn are defined as part of *integrational requirements*.

Design Activities. With design activities we refer to solution crafting, including:

- **Design of the Logical Architecture:** *Information system requirements* in general serve as the essential input for the design of the logical architecture. Within the information system requirements, however, the *domain-specific capabilities* are of major interests, because these define the main functionality to be designed from an architectural perspective. Furthermore, the single *business domains* should be used to guide the initial structure of the architecture into logical components that fits into the future business environment (one component supporting with its functionality all the business processes of one particular business domain).
- **Design of the Technical Architecture:** As *system quality requirements* and *(technical) architectural constraints* have crosscutting character and can directly

10 Note that a further cost estimation is performed with the *business demands analysis*. We refer in this context, however, to cost estimations regarding the development of a system under consideration.

impact on the technical architecture. For example by restricting the used technological infrastructure.

- **Implementation:** *Business restrictions*, including design and implementation guidelines, such as coding standards, can directly restrict the implementation.

Quality Assurance Activities. We concern with quality assurance analytical and testing quality assurance:

- **Analytical Quality Assurance:** Activities within this area aim at discovering deficiencies in order to precociously counteract. *Domain-specic application capabilities* and *system quality requirements* are in the primary scope as one target consists in ensuring the compliance of the growing architecture with the customer's needs regarding quality.
- **Testing Quality Assurance:** Inspections and reviews as performed within quality gates make use of the whole specifications, because they are meant to prove the adequateness of the specifications for a specific purpose such as acceptance. Furthermore, acceptance test cases being one result of testing quality assurance, prove if the delivered information system meets the customers' expectations. Hence, the acceptance test cases are mainly derived at the basis of *information system requirements*, in particular the *use cases* and the *information system services*. A further input for defining acceptance test cases is the *traceability matrix*, as it provides support for checking how many requirements are covered by particular test cases.

3.9. Process Model

We subsequently introduce the basic concepts of the process model for the business information systems' analysis. We first give an overview on the process and describe also the limitations of the process model. Afterwards we describe the steering principles of the process respecting the content production. We define furthermore the milestones of the process and conclude with a description of the tasks to be performed in order to produce the artefact types.

3.9.1. Overview on Business Information Systems' Analysis

As depicted by Fig. 3.31, the business information systems' analysis includes several activities and a specific set of milestones in-between these activities. In particular, we define the activities *customise to project environment*, *create business specification*, *create requirements specification* and finally *change management* activities. Furthermore, we distinguish in general between management activities and engineering activities. The latter concern the production of the introduced artefact types and are described in detail in Sect. 3.9.4 and Sect. 3.9.5. Out of scope are for reasons of simplicity activities that concern the creation and maintenance of the traceability matrix. The milestones and their coupling to specific artefacts are defined in Sect. 3.9.3.

We give now an overview on the activities.

Customise to Project Environment. This activity concerns all the planning tasks that have to be performed and decision that have to be taken during business information systems' analysis. In particular, it concerns how exactly to perform the

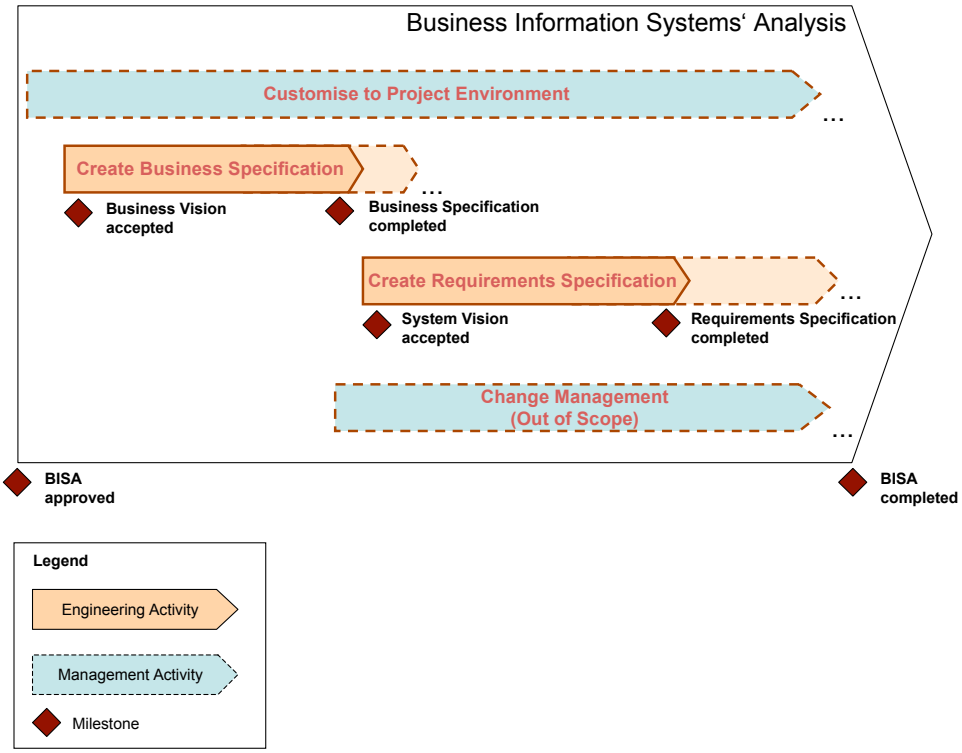


Figure 3.31.: Overview on Business Information Systems' Analysis

engineering activities according to project-specific parameters. In fact, the (engineering) activities introduced in subsequent sections do not dictate any strict order for performing the tasks or how exactly to define the milestones (and their relations). Both are part of the customisation approach that is defined in chapter 4.

Create Business Specification. This activity describes all the tasks that are needed to create the business specification. It is encompassed by two milestones. The milestones define points in time in which specific artefacts are accepted by all parties, respectively in which the artefact type is completed. In particular, the first milestone defines when the business vision is accepted, the second one when the business specification is completed and accepted. Note that the first milestone does not define when exactly to start the activity itself, but only when the first content item (the business vision) is accepted by a customer.

Create Requirements Specification. This activity describes all the tasks concerning the creation of the requirements specification. Similarly to the creation of the business specification, we define also in here two milestones. One that defines the point in time in which to initiate the activity in full (when accepting the system vision) and the second milestones that addresses the completion of the requirements specification.

Change Management. Change management is the nearer sense an own discipline that pervades the whole development life cycle [JBR99, KK03]. It defines tasks for a structured performance of a change request towards selected artefacts and the execution of the change. The latter concerns the creation of new artefacts

or the modification of existing ones. When exactly to initialise change management strongly depends on process model of an organisation. Still, a change can be triggered at any time, i.e. when the first artefact is accepted by a customer. It is for example possible the define additional requirements after accepting the requirements specification. This is also the reason why we continue the depiction of the engineering tasks with a dotted line after their second milestone. However, the change management process is not in scope of this report. Further information regarding change management in general and aspects that concern changes on requirements can be taken from [MFsA09].

Genericness enabling Customisation

The process model is described in a generic, but comprehensive way. Otherwise it would not offer a flexible framework that allows customisation.

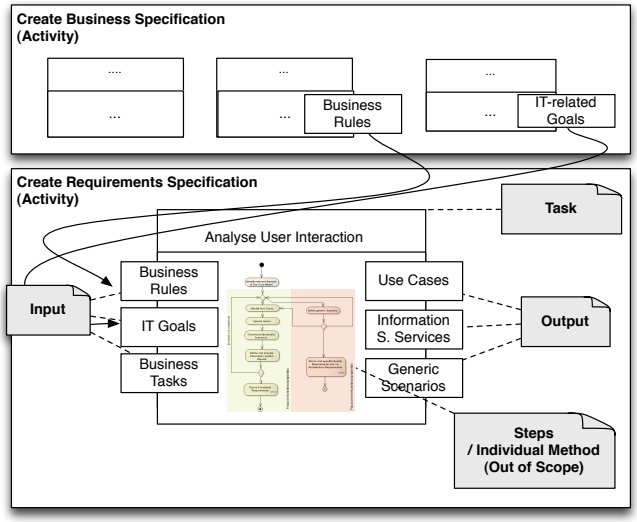


Figure 3.32.: Exclusion of Task Orientation (Methods and their ordered Enforcement)

Fig. 3.32 illustrates an excerpt of the process model that is defined according to the structure described in the meta model of Chp.2. The process model consists of activities and tasks. An activity like *create requirements specification* defines a set of tasks used to produce the requirements specification. Each task in turn produces based on an input an expected output. For producing this output, each task includes steps that define a concrete method for constructing selected contents (output) of the particular artefact type. Methods, however, define for a chosen syntax how exactly to produce the content items, like how to describe a use case with UML activity diagrams.

The process model defines a framework by the definition of tasks and their interdependencies, based on the needed input for constructing the output. Still, it does not restrict the process with specific methods and it does not dictate any specific process description (a sequence of using the methods).

Process Description. The concrete process description is an outcome of the customisation approach, that defines how exactly to act within the process model. A specific process description, such as when referring to a waterfall model, is not

predefined within the process model. It is systematically constructed during the execution of the process at the project level, as introduced in Sect. 4.3.

Methods. Methods including the definition of how to elaborate artefacts with a specific syntax are not defined within the process model for one particular reason: A strict definition of a concrete method for each task excludes the possibility of customising the model to needs at organisational level and at project level[11]. Each organisation defines — or needs at least the possibility to define — selected methods that describe for example how to construct the scenarios of a use case with UML activity diagrams. This is mostly done according to the organisational culture including also the description of best practices.

In Sect. 3.9.4 and Sect. 3.9.5 we define for this reasons a generic *method frame* including a general depiction of the tasks (with the input/output-relation) without defining detailed insights and restricting the syntax. In Chp. 4 we illustrate then the definition of chosen methods at the organisational level.

3.9.2. Steering Principles for the Content Production

The followed philosophy for creating the artefacts can be stated by two steering principles:

1. Problem Statement and Problem Solving is an iterative approach.
2. Each iteration should be performed according to the defined business domains.

Iterative Problem Statement and Problem Solving. According to Conklin et al. [CW97] and as stated in Sect. 3.2.3.3, the elaboration of an idea for a system's solution begins constantly during the elaboration of detailed problems (business and requirements specification). This is especially true as we continuously narrow down the problem space with each design decision that we make. Indeed, with every stated solution, new problems (hidden requirements) arise that would otherwise not be stated by customers as they simply can not have all possible requirements in their mind. This includes also the elaboration of requirements towards information systems and business process descriptions in parallel. For instance, when defining use cases, additional modifications on already specified business processes can be necessary. Similarly, when designing the system, additional modifications on already specified use cases can be necessary. Figure 3.33 illustrates the resulting approach for problem solving (with the dotted lines) in comparison to a traditional view.

In fact, how exactly to proceed strongly depends on constraints on the process models such as implications on a waterfall for example due to a multi-staged bidding procedure (see also Chp. 4). Still, an efficient business information systems' analysis is based on an iterative approach between the problem statement and the problem solving, i.e. the continuous elaboration of a solution and the description of the problems in parallel.

Business Domain-Orientation. For structuring the BISA approach we make use of business domains (see also content item 3.4.2.2 of the business specification). Business domains do not only structure the content of the artefacts, but also the process for elaborating the artefacts. In particular, each of the iterations that are

11 Note that the definition of a concrete method with a specific syntax for one particular concept strongly affects the definition of the methods of surrounding concepts what in turn implies a downslide of the flexibility for using the approach in projects.

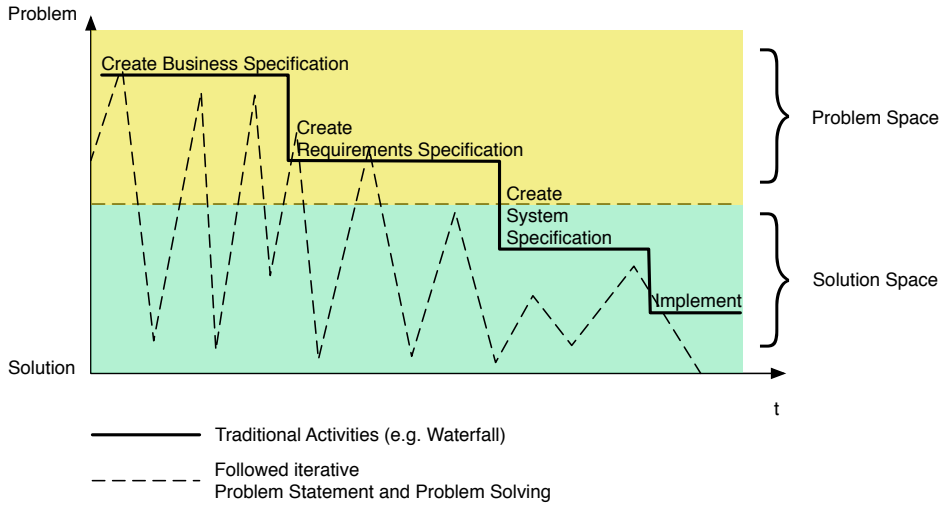


Figure 3.33.: Problem Statement versus Problem Solving according to [CW97]

depicted in Fig. 3.33 reflect the guided elaboration of the content of one particular business domain (see also Fig. 3.34).

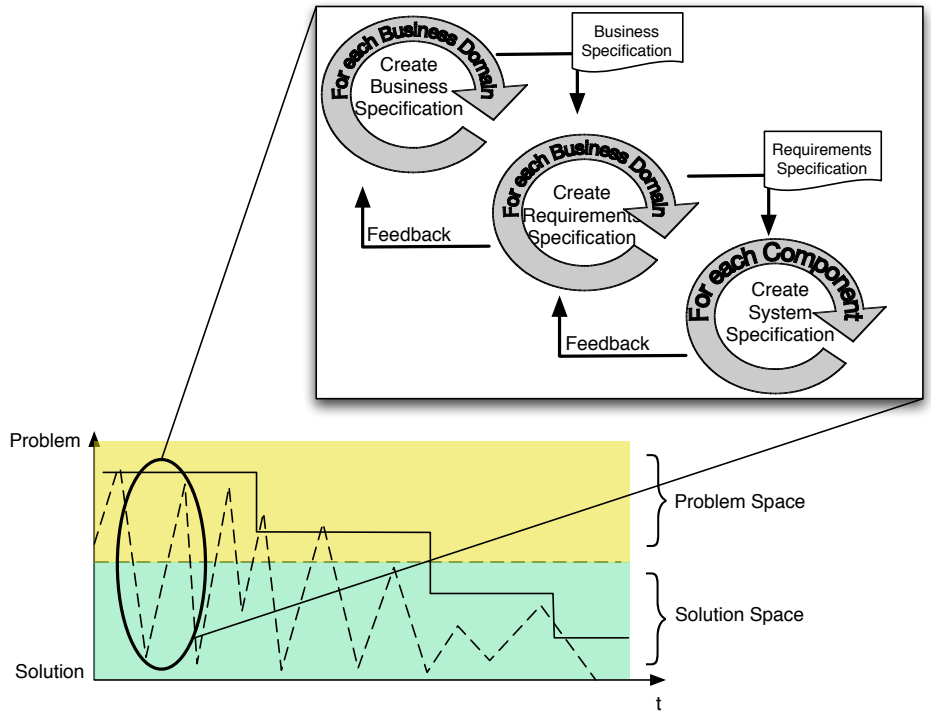


Figure 3.34.: BISA Structured according to Business Domains

Even if there exist dependencies between the business domains, they are used to structure the approach, because they support:

1. a *structured communication with customers*: By grouping the different stakeholder (and process owners with "underlying" user groups) according to their business processes, they serve as a means to support a systematic approval process. For instance, when performing workshops regarding the needs of the roles within single business domains. They furthermore cluster the content of the artefacts into a manageable and well organised amount (according to the business processes).
2. *balanced problem-orientation*: Generally, as described in section 3.2.3.3 the artefacts that correspond to one business domain are refined to detailed workflow descriptions and detailed requirements, but they must not at any cost. As a specific content of the artefacts is related to a business domain, this enables an uncritical differentiation between needed and possible refinement according to project-specific parameters (see also the next chapter 4).
3. *scalability*: Business domains are meant to structure large-scale organisations by their business processes as well as small-scale ones. That supports the scalability of the approach and makes it fit for the use of complex development projects with large application landscapes.

3.9.3. Milestones

We distinguish four basic milestones during the performance of the business information systems' analysis:

1. *Business Vision Accepted* coupled to the content item *business vision* (Sect. 3.4.1). This milestones defines the point in time of accepting the business vision and initiating the activity *create business specification* in full.
2. *Business Specification Completed* coupled to the overall artefact type *business specification*. This milestone defines the point in time of accepting the business specification and thereby setting a reliable basis for the requirements specification.
3. *System Vision Accepted* coupled to the content item *system vision* (Sect. 3.5.2). This milestones defines the point in time of accepting the system vision as a point of entry into the activity *create requirements specification*.
4. *Requirements Specification Completed* coupled finally to the overall artefact type *requirements specification* and defining when to finish the elaboration of the requirements specification.

Figure 3.35 illustrates the four milestones of BISA. It furthermore illustrates the relation of the milestones *business specification completed* to the milestone *system vision accepted* in dependency to the chosen process model. A similar relation is given by the activity *create requirements specification* and design, i.e. when to finish one artefact type and when to start the elaboration of the next.

There exist in general two scenarios for defining these milestones' relations: first, if developing continuously and second if facing a (strict) waterfall model.

Continuous Development. The first possible point of entry for creating a requirements specification (the dotted one) is given if developing continuously, i.e. if the business specification does not have to be completed before accepting the system vision.

This case demands for a minimal (syntactic) completion criterion for the content of the business specification before being able to complete the system vision[12]:

12 Respectively the completion criterion for the content of single business domains (see also the fore going section).

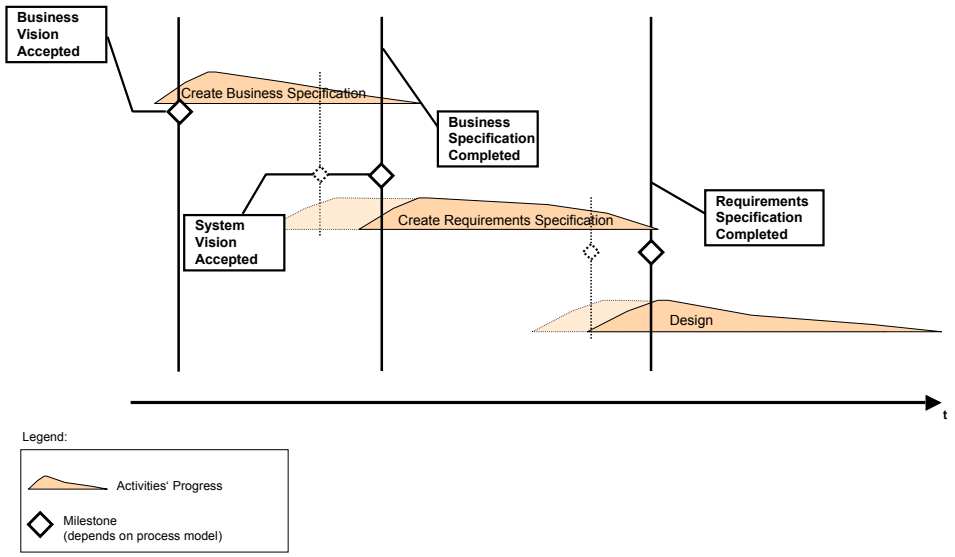


Figure 3.35.: Milestones and their Relation depending on the Process Model

1. The business domains have to be identified.
2. The business services and the business processes should be completely specified in their breadth at the abstraction level of the business process hierarchy. This includes for the core processes the identification of all encompassed business tasks and all participating user groups.
3. The process steps and their relation (describing workflow) do not have to specified completely and can be described during the elaboration of the use case model.
4. All business objects that are directly related to envisioned business tasks have to be identified.
5. The business rules have to be identified, especially the ones affecting the business processes within the envisioned business domains.

At this basis the system vision can be completed by identifying the use cases (according to the business tasks), by identifying the information system services and finally by defining the system overview with e.g. a context diagram.

Waterfall Model. The second point of entry for elaborating the requirements specification describes the elaboration of the system vision when the business specification is specified in full, what comes close to a strict waterfall model. For instance, this can be the case when the results of one activity have to be produced before initiating further going activities, e.g. if we want to clearify if to proceed with the elaboration of the system vision based on the content of the business specification.

Comparison of Both. What process model to chose depends on project-specific aspects (see also Chp. 4). Anyway, the earlier the initial milestone is set, the weaker the system vision. The consequences might be that initial calculations made for pricing (at the basis of the system vision) can exhibit defects in their details. But at the same time, the later the system vision is generated, the more inefficient and cost-intensive the process.

3.9.4. Activity: Create Business Specification

Table 3.2 on page 127 illustrates the tasks of the activity for creating the business specification. The elaboration of the business specification is performed as introduced in Sect. 3.2.3.1 illustrating an engagement roadmap. After defining the current state of an organisation including current business processes and services a gap analysis is performed in order to sketch a vision on the future state. The business goals are decomposed in order to steer the decomposition of the business processes into a customer-adequate direction. In particular, the decomposition is performed for business processes that directly contribute to the satisfaction of the business goals and the envisioned business services (the core processes). Based on the gained ideal business process logic, the structure (the business process hierarchy) is re-designed and adjusted (comparable to a divide and conquer approach).

However, two critical aspects that are to be taken into account, but are out of scope for this report are aspects of service and systems' evolution and consequently aspects of choosing an appropriate service granularity when defining the services. The more abstract a business service is defined (the more business processes "encompassed") the less the gained agility. We exclude then during the evolution possibilities of performing fine-grained analysis of detailed business tasks and thereby analysing the necessary support of business information systems.

Task	Output Artefact	Input Artefact	Description
Define business context	Business context (3.4.1.1)	Business cases, strategy papers, policies, ...	Identify and describe objectives and principles
Derive business goals	Business goals (3.4.3.2)	Business context (3.4.1.1)	Identify business goals at the basis of the mission and value chain
Elaborate restrictions	B. restrictions (3.4.3.1)	Policies, ...	Identify business rules and constraints
Identify current business processes and business services	Business capabilities (3.4.4)	Business cases, ...	Identify as-is business processes (no tasks) and services
Perform gap analysis	Project scope 3.4.1.2	Business goals (3.4.3.2), business capabilities (3.4.4)	Investigate if current state matches with defined principles and identified business goals
Complete business vision	Business vision (3.4.1)	Elaborated content items	Complete based on the defined project scope the business vision and approve it.
Perform stakeholder analysis	Business roles (3.4.6)	State charts, workshops	Identify stakeholder, process owner and user groups related to envisioned business scope
Define process-related goals	Process-related goals (3.4.3.2)	Business goals (3.4.3.2), principles (3.4.1.1)	Define process-rel. goals that satisfy b. goals
Categorise business processes	Business process categories (3.4.4.2)	B. services (3.4.4.1), B. goals (3.4.4.1)	Categorise processes based on how they contribute to achieve business goals and services
Decompose envisioned business processes	Business processes and business tasks (3.4.4.2)	Process-related goals (3.4.3.2), workshops	Define workflow with business tasks for core processes
Define business information model	Business information model (3.4.5)	B. processes (3.4.4.2) and b. services (3.4.4.1)	Derive business and information objects
Define business domains	B. domains (3.4.2.2)	B. roles (3.4.6), b. capabilities (3.4.4), b. objects (3.4.5.1)	(Re-) define business domains
(Re-) design business services and organisation structure	Organisation structure (3.4.2.1), business services (3.4.4.1)	Business processes (3.4.4.2)	Adjust structure according to ideal business processes
Define needs for automatisation	IT-related goals and costs for automatisation (3.4.7)	Business processes and services (3.4.4), business cases, ...	Identify what business services and processes need to be supported by information systems
Rationale decisions	Business demands analysis (3.4.7)	Identified risks, value and cost for each b. service	Complete business demands analysis
Specify glossary	Glossary (3.4.8)	Workshop protocols	Define domain-specfic terms in glossary

Table 3.2.: Overview on Tasks of the Activity *Create Business Specification*

3.9.5. Activity: Create Requirements Specifiation

Table 3.3 on page 129 illustrates the tasks for creating the requirements specification. Same as done with the tasks concerning the business specification, we do not define any order for performing the tasks.

However, the first depicted task is the one that concerns the definition of the system vision. The system vision can be defined when the business specification has a sufficient degree of completion, i.e. the business tasks are (initially) elaborated. These business tasks serve as an input for defining the scope of the envisioned system. When the system vision is defined the content is elaborated for each of the business domains. This especially includes information system requirements. This elaboration is gained by the performing for example workshops for the stakeholders that relate to the corresponding business domain (see also Sect. 3.9.2). Within these workshops, requirements are

1. elicited (harvested). Appendix A.3 introduces for this purpose a template that facilitates the initial elaboration of requirements.
2. validated and refined
3. classified into corresponding types, respectively "transferred" in dependency to their assertion into a proper shape that corresponds to the elements of the concept model

In the primary scope are mostly interaction models, including use cases, generic scenarios and corresponding actors ("what is done by whom?") and finally information system services. In dependency to those artefacts, the information system objects can be identified and specified.

Finally, each requirement should underlie a validation ensuring a certain quality (of the artefacts). During validation the requirements are checked for specific quality criteria and defects, such as checking them for (technical) feasibility (see also [oEEE98]). This quality assurance is performed from a constructive view (when creating the requirements specification) as we recorded that many quality assurance tasks need an insight into the projects content that can not be gained when checking a specification within a quality gate. An example would be to check a sufficient correctness of requirements that often is not comprehensible to third parties [SEH09]. For the same reasons, upcoming risks are also calculated during this validation. Both aspects serve as a basis for (re-) adjusting the priority of the requirements according to their risks and finally their potential costs (for implementation). Regarding the latter, it has to be clear that an exact cost calculation in early stages is impossible, because the costs are influenced by too many (e.g. technological) factors. Still, an approximation based on function points, respectively use case points, is possible (and advisable) by referring to the use cases and the generic scenarios.

Task	Output Artefact	Input Artefact	Description
Gather business needs	Summary of business specification (3.5.2.1)	Business specification	Summarise content of business specification being in scope of the system
Scope system's capabilities	Use case overview (3.5.2.2.1), IS. service overview (3.5.2.2.1)	B. services and b. tasks ((3.4.4)	Identify candidates for use cases and services
Scope system's boundary	S. overview (3.5.2.2.3), ext. systems (3.5.2.2.4)	—	Define interrelation to operative environment
Complete system vision	System vision (3.5.2)	—	Complete vision (e.g. by additional features) and align with business specification
Harvest and prioritise requirements	abstract requirements	—	Elicit initial requirements and prioritise by business value (e.g. by workshops)
Analyse user interaction	Use case model (3.5.3.3.1), actors (3.5.3.1), information system service model (3.5.3.3.3), Generic scenarios (3.5.3.2)	Business capabilities (3.4.4), business rules (3.4.3.1, IT-related goals (3.4.3.2), workshop protocols	Define expected (functional and non-functional) user interaction and services
Derive functional requirements	Functional requirements (3.5.3.3.2)	Use case model (3.5.3.3.1), workshop protocols	Define functional requirements
Define information system objects	Information system objects (3.5.3.4)	IS services (3.5.3.3.3), use case model (3.5.3.3.1)	Define processed IS objects in relation to information objects
Quantify system quality requirements	System quality requirements (3.5.3.5)	Gen. scenarios (3.5.3.2), IT-related goals (3.4.3.2), QoS (3.5.3.3.3)	Define system quality with appropriate measurement
Define architectural constraints	Architectural constraints (3.5.3.6)	Generic scenarios, business constraints, IT-related goals	Define logical and technical architectural constraints
Define organisational requirements	Organisational requirements (3.5.5)	Workshop protocols, bidding documents	Define project requirements and obligations
Define integrational requirements	Integrational requirements (3.5.4)	Project plan, information system objects (3.5.3.4)	Define deployment and migration requirements
Validate requirements and assess risks	Requirements risk status report (3.5.6)	Any requirement	Check requirements for defects and assess risks
Adjust priority	Any requirement	Any requirement	Adjust priority based on risks and costs
Extend glossary	Glossary (3.5.7)	Workshop protocols	Define technical terms

Table 3.3.: Overview on Tasks of the Activity *Create Requirements Specification*

3.10. Role Model

We distinguish within the role model between primary roles with certain responsibilities on selected artefact types and indirectly participating roles.

3.10.1. Primary Roles

During business information systems' analysis we define two primary roles: The *business analyst* and the *requirements engineer*. Although we focus on roles that take the responsibility for chosen artefacts, further team members can be included into the roles model such as by distinguishing between a chief business analyst and further business analysts.

Business Analyst. The business analyst is characterised by a particular domain knowledge concerning of selected (industrial) customer domains like the financial sector or insurance and resulting business process structures within such domains. He has the responsibility on the creation and maintenance of the business specification and is for the customer the primary contact for any discussions and decisions regarding the content of the specification. The business analyst is the key roles concerning internals of a customer's organisation and the business.

Requirements Engineer. The requirements engineer is characterised by aligning the business needs with the needs towards underlying IT systems. The requirements engineer takes the responsibility on the creation and maintenance of the requirements specification. 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 terms of being a target group of the requirements specification. He takes furthermore the responsibility for the fulfilment of the requirements and thereby the traceability matrix.

3.10.2. Indirectly Participating Roles

There exist additional roles like the test manager or risk manager. However, we do not define these roles within the role model, because these roles have to be provided by the process model into which the introduced core model is integrated (at the organisational level, see also the next chapter).

To give a brief overview, further roles that indirectly participate in the process can be:

- System architect, possibly separating the role of the architect according to the architectural layers (the logical and the technical one)
- Project leader and project manager concerning the overall project execution (also including release planning activities) and contract management and taking the responsibility for the like
- Risk manager concerning the analysis and the mitigation of any kind of risk factors
- Quality manager concerning the control of inspections, reviews of defined specifications (as part of quality gates) and further artefacts that need quality assurance
- Test manager that is defined in addition to the quality manager concerning the planing, definition and execution of test cases

Roles like the introduced ones make use of the contents of the business specification and / or the requirements specification, as they need selected artefacts for the elaboration of the artefacts they take the responsibility for. For instance, the risk manager uses the requirements risk status report as an entry point into the risk factors that arise from the requirements specification. Further information can be taken from Sect. 3.8 that shows what artefacts serve as an interface into the development life cycle.

4. Customisation and Integration Approach

We describe in this chapter the customisation and integration approach. It customises the artefact-based approach of chapter 3 to:

1. *Organisational environments* concerning a process integration into environments of individual organisations.
2. *Project environments* concerning the adaptation of BISA efforts according to individual project parameters in order to enable a custom project execution, i.e. a balanced problem orientation as introductory described in chapter 1.

In section 4.1 we introduce first the different stages of customisation. We define afterwards for each of the stages the principles and basic mechanisms. We show furthermore how individual best-practices of particular companies can be integrated.

In particular, we define in Sect. 4.2 the mechanism for the first customisation stage. In Sect. 4.3 we introduce the customisation to project environments. The process integration into a process framework (the first customisation stage) is illustrated at the example of the V-Modell XT. The V-Modell XT (short: V-Modell) is a development process model, which is based on a meta model focussing on artefact-orientation. It is a process framework that supports variability in many areas. It's concrete reference implementation is the standard development process for IT-projects within the German government [FHKS08].

Contents

4.1. Stages of Customisation

There exist several understandings of the term *customisation*, mostly depending on the research area in which the term is used. Heavily used terms are also *tailoring* and *pruning* concerning for example the deletion of specific pieces of a model in order to adapt it towards particular needs of for example a project. An overview of different terms and corresponding approaches in which these terms are used is given in Chp. 1.3.

What terms exactly are used strongly depends on the concern of customisation. Some approaches consider the customisation in the sense of process integration, other approaches in the sense of customisation before or during the execution of a project. In any case they consider the customisation of a model according to individual needs in terms of modifying it, creating and / or deleting parts of it, why we explicitly use the term *customisation* (instead of e.g. tailoring).

This customisation is performed over several stages [KH08], each of the stages reflecting different abstractions of customisation. Figure 4.1 gives an overview.

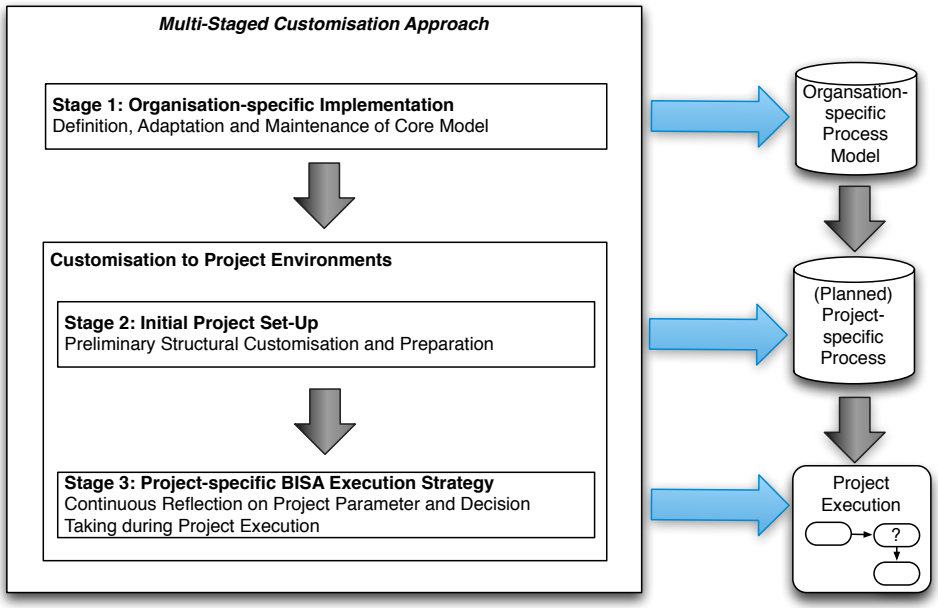


Figure 4.1.: Overview on Different Stages of Customisation

We distinguish in general between *organisation-specific implementation* (described in Sect. 4.2) and *customisation to project environments* (described in Sect. 4.3). The first considers the (process) integration of the artefact-based core model into organisational environments. The second considers the inference of a project-specific BISA process being conformant to the reference model that is introduced the chapter before. Regarding the customisation to project environments we distinguish between the set-up of a process frame before the execution of the process and the fine-grained decision taking within the process frame during the execution of the process[1]:

- *Initial project set-up* describes the preparation of the core model according global parameters that are known from the beginning of a project. The approach at this stage affects the milestones, the creation of the specifications

1 Both stages are often referred to as static or structural tailoring and dynamic or content tailoring.

structure and the roles. Hence, it defines the frame for the process model at the project level.
- *Project-specific BISA execution strategy* describes during the project the continuous reflection on project parameters and a systematic decision taking within the pre-defined process frame towards the necessary and possible degree of problem-orientation.

In the following, the different stages are defined in detail.

4.2. Organisation-specific Implementation

The approach aims at the systematic integration of the core model into existing process models of a particular organisation and the definition of methods that are chosen according to an individual organisational culture. After the integration of the core model, it serves as an integrated reference approach for individual projects.

The organisation-specific implementation, however, is an essential step that is performed once before applying the core model at project level. Otherwise, the interdependencies of the BISA approach to further development and management activities of the envisioned organisation would for example not be clear, affecting negatively a continuous and efficient process.

We first give an overview on the approach and define afterwards the steps to be performed. Wherever reasonable we already refer to the V-Modell since this framework will be used in Sect. 4.4 as the exemplary reference model for the process integration. An introduction into the V-Modell and its meta model is given in appendix B.

4.2.1. Approach Overview

The integration is performed over three basic steps. Figure 4.2 depicts the approach that consists of:

1. An analysis of existing process models of an organisation (including content and structure) and contemporary needs as a preparation for the integration.
2. The customisation of an excerpt of the process model, i.e. the modification and extension of the model according to the artefact-based core model.
3. The integration of the modified process model into the organisation for further usage at the project level.

4.2.2. Analysis of the Process Model and Needs

Before implementing the approach for BISA into a process model, an analysis of this process model has to be performed. The purpose of conducting such an analysis is to investigate the possibility, but also the necessity of implementing a new approach into a particular organisation. On the one hand, this includes the investigation of contemporary needs in terms of costs and benefits respecting early stages of development. On the other hand, the analysis includes the preparation of the integration itself, for instance by determining the strategy for integrating the core model into the organisation.

In any case, the analysis is mostly performed as an own project. It is conducted over four steps, including:

1. An analysis of the process model at the organisational levels

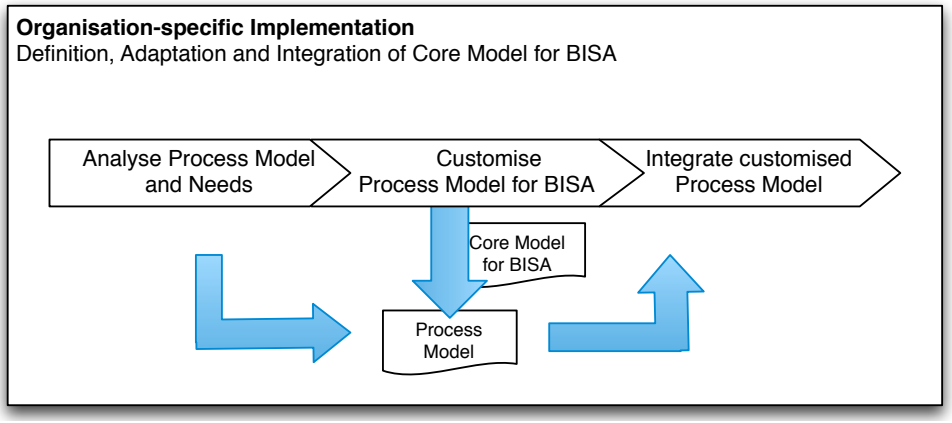


Figure 4.2.: Overview on Organisation-specific Customisation

2. An analysis of the process model at the project level
3. An analysis of the contemporary needs
4. A proposal of an integration strategy

4.2.2.1. Analysis of Process Models at Organisational Level

The first step is performed by analysing the description of the processes and the methods at an organisational level (question: "How is the process defined?"). The study objects are in particular descriptions of reference processes, existing reference role models, artefact models and finally material of (internal) training courses. Most organisations define their processes as part of guidelines or standard operating procedures that adapt existing approaches like the RUP to internal needs. The most important prerequisite for the integration is that the process model of the organisation exhibits a modular structure that allows the modification of chosen parts such as by separating roles and artefacts from processes. Obviously, the model should be based on a meta model or at least one should be able to derive a meta model from the process descriptions (as it is the case for RUP).

V-Modell XT as a Process Framework

For the integration we use a standard V-Modell as basis. Using the newly introduced variability concepts we provide an extension model, which contains the demanded modular structure. Furthermore, the V-Modell offers variability operations[2] to customise the reference model.

4.2.2.2. Analysis of Process Models at Project Level

This step aims at understanding how the process is documented and lived in projects (question: "How is the process performed within projects?"). This investigation supports

- discovering mismatches between the reference process of the organisation and the needs of individual projects.
- gathering typical project parameter that reason the mismatches between the reference process and the individual project execution, i.e. parameters that

2 In addition to the standard V-Modell XT we use an extended meta-model that contains a richer set of variability operations. It is a superset of the standard meta-model an includes all standard operations.

affect the ability and necessity of elaborating specific artefacts according to the existing reference process (see also Sect. 4.3).
- understanding the culture of organisations reflected by used (preferred) methods at the project level for producing selected artefacts.

The analysis can be performed by content analyses of produced specification documents and structured interviews. The specifications can be analysed by comparing the produced content with the artefact model (see the fore going chapter).

This comparison supports the detection of missing artefacts what in turn serves as a basis for structuring the interviews with corresponding (chosen) project participants in which one can distil initial project parameters. Research questions that can guide this analysis step are "to what extent are artefacts documented?" (achieved by the content analysis), followed by the question "what project parameters take effect on the elaboration of the artefacts?" (achieved by the structured interviews).

4.2.2.3. Analysis of Contemporary Needs

An investigation determines in terms of business cases if the integration makes sense at all and in case what details are important regarding the integration (question: "What are the costs and the benefits of the defined process?"). A way of analysing costs and benefits could be to compare projects that performed a kind of requirements engineering (representing projects in favour of an integration) with projects that performed a purely solution-oriented process (representing projects being not in favour). Indicators for this comparison can for example be the process costs calculated using activity-based costing. In addition to the process costs, further indicators should be taken into account that give hints on the total economic effects of those efforts, i.e. that show if the efforts paid off.

4.2.2.4. Proposal and Approval of an Integration Strategy

The last step of the analysis should be to present the results to the participants of the analysis and to clear open issues in a discussion (question: "Shall the existing process model be extended and integrated for BISA and if yes, how?"). This presentation should also include a proposal of an integration strategy that defines how to integrate the artefact-based core model for business information systems' analysis (see also the last step of the organisation-specific implementation in Sect. 4.2.4).

4.2.3. Customisation of the Process Model for BISA

Once, the customisation of the process model at organisational level has been approved, it has to be performed. This customisation includes the modification and / or the extension of existing parts of the process model. In order to perform this customisation, a clear excerpt of the process model has to be defined.

4.2.3.1. Definition of an Excerpt of the Process Model to be Customised

The results of the analysis of the projects at the project level (see the section before) support the identification of typical scenarios. These scenarios reflect performed projects including the definition of milestones, the production of artefacts and the use of particular methods. These project scenarios should in general aim at a (re-) design of a business process landscape and / or a RE as part of a development of custom software.

V-Modell XT: Process and Tailoring Model

The V-Modell offers two basic concepts that enable the definition of (concrete) project scenarios: The process model and the tailoring model.

Process Model. Via the assignment of single work products to decision gates, the product model couples work products to process steps. Decision gates are the interface for the connection of the static work products and the dynamic process.

To embed the decision gates into the process it is required eiter to use a decision gate type already referred in an existing procedure module or to create a new procedure module which refers to a new or existing decision gate type.

As described, for the mapping of BISA it is necessary to define the possible areas of application first. After this it can be decided which existing processes might be suitable and which new processes have to be created. If existing process match the requirements, no new procedure modules have to be created – only the new work products have to be assigned to existing decision gates. As a consequence, they are automatically embedded in the process. If there are new processes identified, new procedure modules have to be designed. In that case, the customisation of the tailoring-profile of the V-Modell has to respect the new process modules as well.

As BISA only defines some abstract process elements, given process elements of the V-Modell are reused as far as possible. Only the project stage of creating the *Business Specification* is more compex than the comparrable stage in the standard. Hence, the corresponding procedure module has to be created new.

Tailoring Model. As mentioned above, an analysis of the possible areas of integration must be performed to identify suitable integration scenarios. The analysis states: BISA can be applied in projects of type *System Development (Customer)* and *System Development (Customer/Supplier)*. Hence, new process modules and new procedure modules have to be considered and the project types (an comprised type variants) have to customised.

So the V-Modell with BISA extension will support *System Development (Customer)* and *System Development (Customer/Supplier)*. Furthermore, the BISA model will avoid process contents, which have no direct association to software development. While customising the project types, all elements not defined as elementary for BISA will be scrachted from the BISA extension.

4.2.3.2. Customisation of the Process Model

The identified excerpt of the process model is extended (or modified) by:

1. extending the result structure according to the artefact model of Sect. 3.3.
2. adapting the roles according to the role model of Sect. 3.10 (by defining new ones or by extending abilities and responsibilities of existing roles).
3. extending the process model including the definition of a phase, activities and milestones.
4. defining methods in terms of selecting the preferred syntax and task descriptions gathered from the fore going analysis of different projects as a default recommendation for projects.
5. re-setting connectors between the artefacts, the tasks, the milestones and finally the roles.

Regarding the extension of the result structure it is up to first modify the targeted artefact structure (i.e. the chapters) and then integrate the concept model into the product descriptions. In case of a process-based approach, the artefacts must be

in a first step identified by the corresponding (producing) activities that are performed within the defined scenarios.

When defining default methods (reflecting experiences at project levels of representative team members), it has to be clear that the syntax choice for representing one concept type takes strong effect on the possibilities of choosing the syntax for further (directly relating) concept types. For this reason the syntactic consistency of the overall artefact model has to be manually checked on the basis of exemplarily described specification documents.

Finally, when re-setting the connectors, the establishment of the associations affects also the associations between the BISA artefacts and further activities of the development life cycle that are related to the artefacts (going beyond the excerpt of the project scenario). For instance, by taking into account the relations of concept types and content items to the area of the logical and technical architecture design that makes use of information system requirements. Or by taking into account to project management activities that make use of the system vision for performing cost estimations . Section 3.8 defines the relevant artefacts that have to be taken into account as potential interfaces to the surrounding development life cycle.

V-Modell XT: Associations to Management Issues

The extension of the concrete process model of the V-Modell is not stand-alone but needs an integration into chosen contents that are related to (project) management. The management is relevant since it builds the frame for all contents of the process including work products of decision gates.

Management in the V-Modell XT. The management capabilities of the V-Modell is one of its most important strenghts. It defines a variety of work products for project management, including several sub-disciplines such as configuration management, change management or risk management. Furthermore, all management artefacts are embedded into the process (organisation) with links to decision gates. They are also tightly integrated into the product dependency network.

The assignments and dependencies guarantee that management is seamless organised. The integration of BISA has to extend given concepts.

Management Artefacts. On top of the work product structures, some management processes are also defined respecting the special needs of the artefact-based approach. So, artefact-based controlling is established in a V-Modell project using state machines. For each work product a state is taken and traced over its whole life cycle. The state machine includes quality assurance (self tests and independant quality assurance). The V-Modell requires all work products neccessary for a particular decision gate to be in a finished state. This means that the considered work products have to be passed by quality assurance measures. In addition, all finished and dependant work products have to be consistent to each other.

Integration BISA artefacts into the V-Modell-structure also requires to adapt the management philosophy of the V-Modell. State machines have to be introduced and dependencies have to be established.

4.2.4. Integration of the Customised Process Model

There basically exist two ways of integrating the customised process model into the organisation:

1. By a “Big Bang Integration” that announces the new approach and enforces its immediate use within the organisation

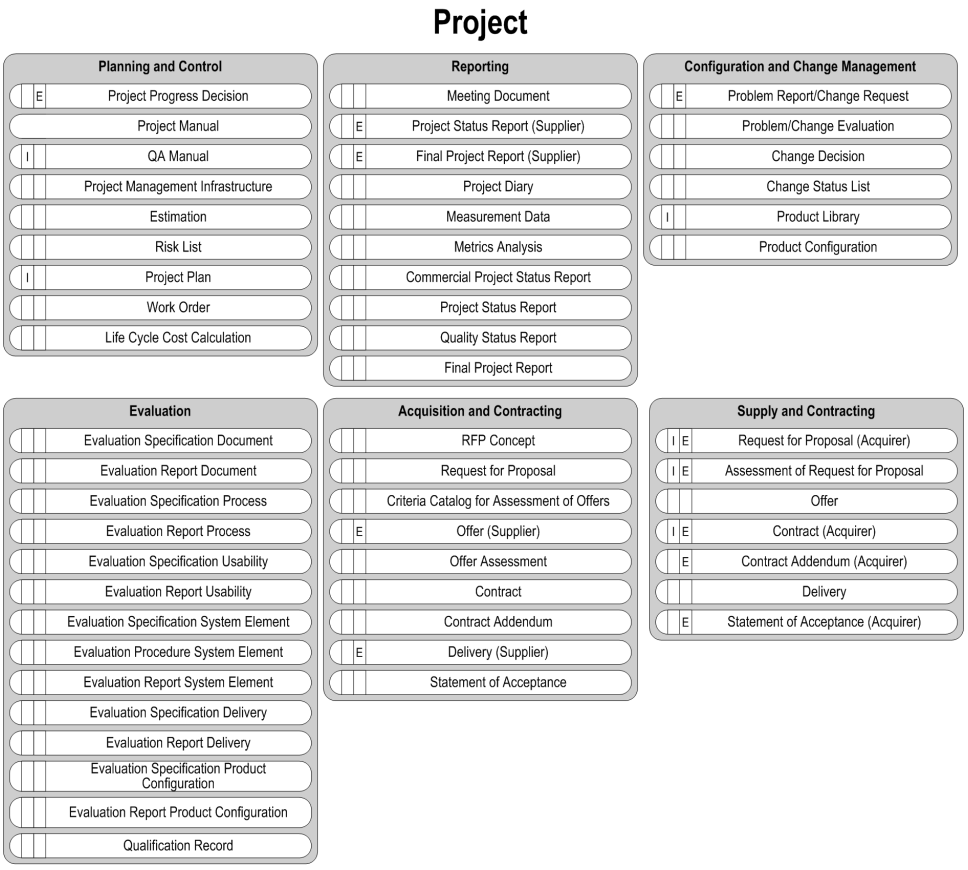


Figure 4.3.: Work Products from Management (Overview)

2. By a step-wise integration with a pilot phase

The step-wise integration is elaborate but supports the acceptance of using the process at the project level. In parallel, the process should be piloted at chosen projects (by actively guiding this pilot). This supports the detection of remaining deficiencies regarding the needs of individual projects. After integrating the feedback of the projects into the process and internally publishing the experiences, the process should be integrated. This integration includes the training of the content in courses and finally an update of the organisational process definitions (guidelines, standard operating procedures, ...) declaring the use of the new process as obligatory.

4.3. Customisation to Project Environments

Once the artefact-based core model is integrated into an organisational environment it has to be customised for each project. This customisation concerns the systematic decision taking during set-up and execution of an integrated process achieving individual project needs. The customisation approach efficiently incorporates individual experiences, needs and best practices into the process and aims at:

1. systematically guiding through the elaboration of the project-specific artefact model respecting decisions towards the necessary and possible degree

of solution orientation (see also Sect. 3.2.3.3).

2. ensuring during customisation (independent of taken decisions) traceable results and thereby continuity within the process chain (i.e. conformance of the models being produced to the referenced artefact model).
3. reflecting the experiences of team members and the culture of an organisation giving the necessary degree of freedom for making decisions.
4. achieving transparency within a repeatable BISA process, i.e. by achieving reproducible decisions and finally reproducible specification documents since the made decisions and their outcome (the documents) have to be comprehensible to external parties (as found in quality assurance activities).

4.3.1. Overview of the Approach

As introductory described, the customisation to project environments is performed over two stages. The first stage addresses the preparation of the business information systems' analysis, the second one the execution. Both stages are performed in dependency to project parameters reflecting with their values individual project characteristics. Obviously, these project characteristics that steer the process are based on experiences of an organisation respecting different project types.

We make for this reason use of a project characterisation into project types and a performance of a project, both according to experiences that are reflected by a knowledge repository.

Figure 4.4 gives a summarised view onto the approach.

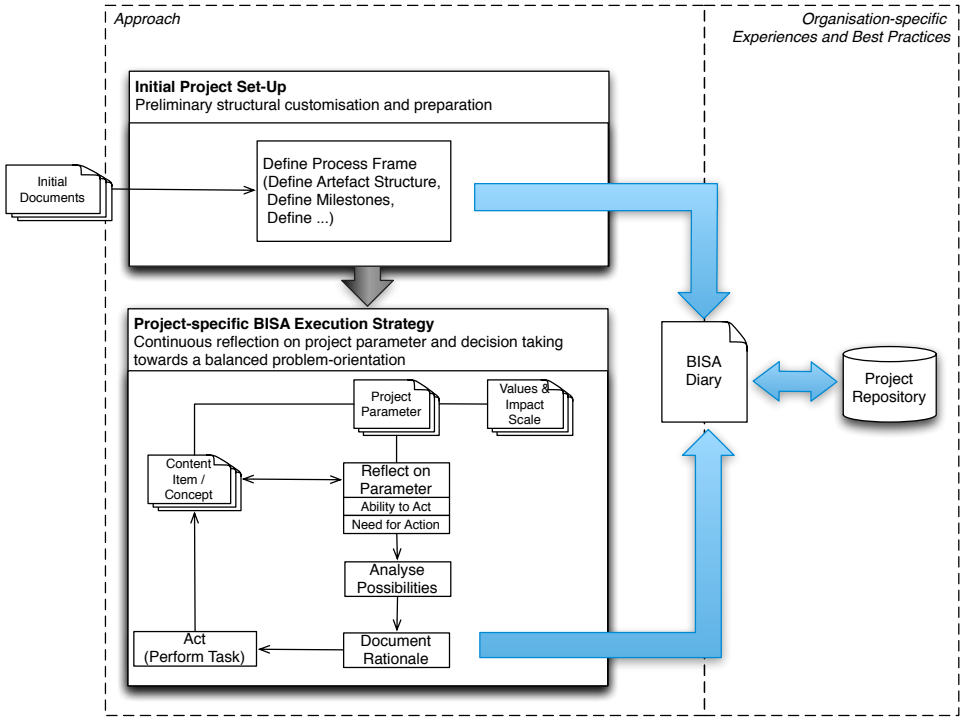


Figure 4.4.: Overview of the Customisation to Project Environments

The left side of the figure depicts the two stages of the approach:

1. The *Initial Project Set-Up*, defined in detail in Sect. 4.3.2, considers the preparation of a *process frame* (process structure) by defining for example milestones and the initial structure of the artefacts.
2. The *Project-Specific BISA Execution Strategy*, defined in detail in Sect. 4.3.3, systematically supports then the step-by-step construction of the process within the given process frame. It defines how (and if) to construct the *content of single artefacts*. It supports the decision on whether to perform a solution-oriented or a problem-oriented process.

Both stages are performed according to selected project parameters that characterise a project. Figure 4.5 depicts on the left side elements of a projects and its characteristics. The right side of the figure illustrates an abstract view on the elements of artefact-orientation. In-between both the connection is given via impacts and impact scales of the parameters onto the elements of the development process.

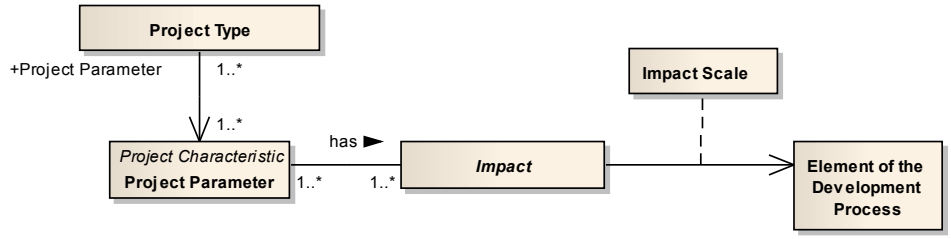


Figure 4.5.: Project Parameters characterising a Project

More concrete, we see:

- A project type being characterised by a set of project parameters (with concrete values).
- Project parameter as assessable circumstances of a project (like the availability of end users).
- The impact of a parameter defining what elements of the development process are affected (like the availability of end users impacting the definition of a use cases model, or information system service model, ...).
- An impact scale showing how the project parameter take effect (like affecting the ability to act and the need for action when producing above mentioned contents).
- Affected elements of the (artefact-based) development process including roles, milestones, activities and tasks, the artefacts' structure and finally the artefacts' content

A concrete project (of a specific type) is therefore characterised by the overall process and its results in relation to specific project characteristics, i.e. a set of project parameters and their impacts.

Hint: *Individual Aspects regarding Project Characterisation*
The project parameters and their impact are valid for all situations, independent of the organisation in which the projects are performed. Instead, the impact scales incorporate individual experiences and practices of single organisations. Similarly, how the project types are defined depends on individual aspects of a particular organisation and the used process framework since there is no common understanding on what exact project parameters belong to a specific project type. Large-scale projects can be characterised by several project parameters like functional complexity, team sizes or efforts in man months. Some process frameworks might define distributed development projects as a specific type, others might define the project types according to possible process models (like agile development projects).

Hence, for customising the artefact-based approach at the project level, the approach supports the decisions to be made by defining what project parameters take effect onto the process elements, the roles, the artefacts' structure and finally during the second stage onto the elaboration of the artefacts' content. How the effect is to be handled, i.e. what decisions to take, reflects best practices in terms of experiences made within other projects of the same organisation. For this reason we make use of a *project repository*, as illustrated on the right side of Fig. 4.4. Each project documents and validates individual experiences within a proposed *BISA Diary* including what decisions have been taken according to selected parameters. The projects persist then in the repository their experiences (reflected by the diary) for further use in other projects that exhibit same characteristics, similarly as known from decision support systems.

Note that an initial set of (individual) project parameters needed for setting up the repository is given from the analysis of existing projects as described in Sect. 4.2.2.

We subsequently give a short overview on the two stages, before illustrating a possible structure of the BISA diary.

Initial Project Set-Up. The stage concerning the project set-up initially characterises the project and aims at:

1. the definition of the overall project scope
2. the structural preparation of the BISA approach for the project execution

The main input for this stage are given documents such as such as bidding documents of customers.

Based on this input and after defining the project scope (after initially preparing the business vision, respectively the system vision) we define the process frame. This includes the preliminary definition of the initial structure of the artefact types to be delivered and eventually the exclusion of chosen content items, i.e. the preparation of the documents and their chapters. The initial content is defined by classifying the content of given documents (requirements specifications, business process descriptions, etc.) and allocating it to the content items respecting the definition of the concept model. Finally, the roles are allocated to chosen team members and the milestones are defined.

Project-Specific BISA Execution Strategy. The stage concerning the BISA execution strategy guides then according to individual project parameters through the elaboration of the artefacts' contents. The approach shows what project parameters take effect onto the need for action and the ability to act respecting the definition of the artefact's content rather then dictating a strict process (a process is implicated since the tasks are coupled to envisioned content items).

As many project parameters arise from customers' domains and vary from business domain to business domain, the approach differentiates between decision on if to specify the content in full or if to leave certain aspects underspecified for each business domain. The approach supports in other words the necessary and possible intensity of BISA by guiding through the decisions if to perform a problem-oriented approach or not, specifically for chosen artefacts' contents that are related to the single business domains.

For this purpose, the approach defines concrete stages of decisions according to the artefact abstraction model (see also Sect. 3.2.3.3). For each stage we allocate project parameters to specific artefacts, respectively content items. The repository shows what decisions have been taken in other projects. For instance, if one project faces the situation that the end users are not available within the business processes of one particular business domain, the customisation approach gives support as it

shows if other projects left in same or similar situations the use case model underspecified and what consequences other projects had to face.

BISA Diary

The BISA diary encompasses all the experiences made during the business information systems' analysis of a particular project. It aims at transparently documenting project characteristics and the corresponding decisions that have been made and serves finally for following projects as a basis for comparing characteristics and for making own decisions.

The diary, however, should be structured in a sense that allows to persist the content of the diary after the project execution within the project repository.

Figure 4.6 depicts an overview on the possible structure of the diary. According to the stages of customising BISA efforts to project environments, we distinguish within the diary between two major content items:

1. the *project setting* that is mostly defined during initial set-up of the project according to project characteristics that take a comprehensive impact onto the overall project (see Sect. 4.3.2).
2. the *project-specific BISA execution strategy* that continuously captures the decisions that are made during project execution and that are rationale for the production of specific contents (see Sect. 4.3.3).

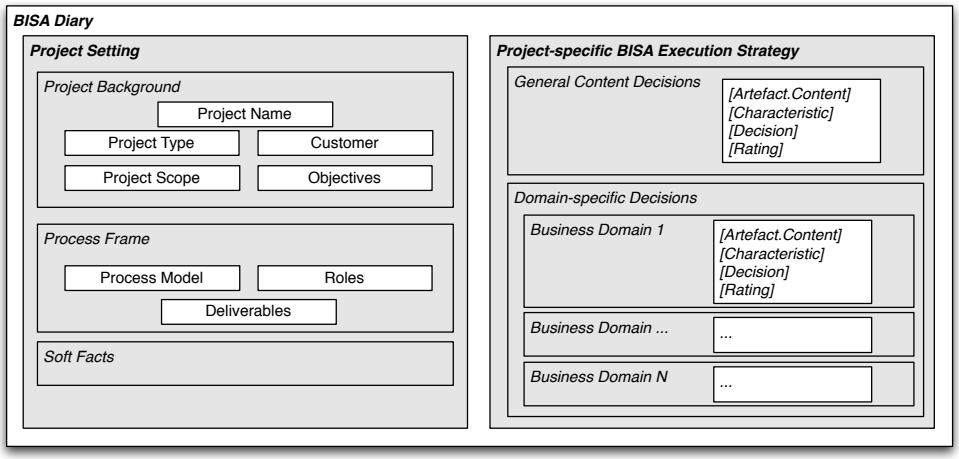


Figure 4.6.: BISA Diary

Project Setting. The project setting characterises the project mostly during initial project set-up according to three aspects: The project background, the chosen process frame and further soft facts.

The project background gives information about the customer, his objectives (respectively business goals) and the project scope. Furthermore, it defines the project type that is chosen according to a set of parameters like "consultancy project" (see the section before).

The process frame defines the roles allocated to concrete team members, the process model including activities, milestones and their relation and finally the deliverables. The latter describes the artefact types and their structure to be delivered. For instance, if performing a development project without the need of a business

specification and in addition agreeing on a lightweight specification, the deliverable could exclusively describe the resulting structure of the requirements specification.

Finally, the soft facts define parameters that have a comprehensive impact onto the project as a whole. They cannot be allocated to e.g. concrete artefacts but still are remarkable occurrences and experiences that additionally characterise a project[3]. Examples can be:

- Resources (budget, technical infrastructure, ...)
- Motivation and experiences of team members
- Relation to external suppliers
- Experiences with stakeholders.

Project-Specific BISA Execution Strategy. The project-specific BISA execution strategy characterises the project according to the results being produced as an outcome of continuously customising the content of the artefacts according to project parameters that arise during project execution. It describes the decided degree of solution-orientation and the reasons for the decisions. We distinguish between decisions that can be allocated to chosen business domains of a customer's organisation and general decisions. For each decision, the diary includes

1. The affected artefact
2. The project characteristic (the parameter and the concrete value)
3. The decision that has been taken according to the characteristic
4. A rating of the decisions, i.e. the consequences that single decisions took[4]

For instance, within the entry "Business Domain Sales", one documented decision could be not to document use cases for the business tasks of the business domain because the user groups were not available for any discussion. An example for an entry within the general (domain-unspecific) content decisions could be to document a high variability of system quality requirements, because of a missing quality consciousness of the stakeholders.

4.3.2. Approach for Initial Project Set-Up

In the following we give an overview of the approach for initially setting up a project. Figure 4.7 depicts the main steps to be performed when setting up the project.

Based on given documents, the project background is elaborated. In a second step, the process frame is defined in which the second stage of customisation constructs the artefacts contents (see Sect. 4.3.3). Each decision that is taken within the set-up of a project must be performed in close collaboration with project management since the decisions take effect on the overall project execution.

4.3.2.1. Elaborate Project Background and Initial Documents

Based on initially given documents the project background is elaborated. Exemplary documents can be:

- Bidding documents and acquisition material including for example project proposals, initial sets of requirements, constraints towards the process model, coarse (planned) time constraints or release strategies
- Existing (frame) contracts if the project is a followup project

3 Soft facts are described for example within the V-Modell as part of the "project diary".

4 The rating of the consequences can only be performed after a project execution. See also Sect. 4.3.3 introducing the approach.

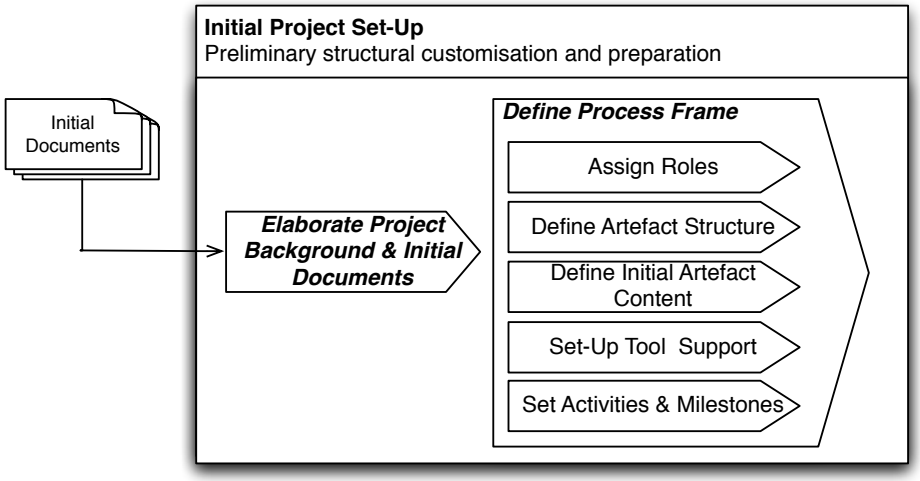


Figure 4.7.: Overview on Initial Project Set-Up

- Requirements specifications, system specifications and test cases
- System-specific guidelines and manuals of fore going (system) maintenance phases
- Organisation-specifif guidelines, policies or standard procedures

Based on these and further information that are known from the beginning, such as the stages of a bidding procedure and existing project requirements, the content is prepared and the background information are documented within the BISA diary.

4.3.2.2. Define Process Frame

In dependency to the project scope and the customer-side bidding procedure, different activities can be assigned. Thus, a project can have many facets and:

- aim at the exclusive elaboration of the business specification, e.g. as part of a consultancy project including (re-) design of the business processes
- aim at the exclusive elaboration of the requirements specification
- aim at both the elaboration of the business and of the requirements specification including (optionally) subsequent system specification(s), ...

Independent of the affected excerpt of the artefact model to be produced, the artefact-based core model (artefact model, role model, process model) is preliminary instantiated at the basis of the scope. Figure 4.7 depicts for this purpose on the right side the steps to be performed, including the:

1. assignment of the roles.
2. definition of the artefacts' structure.
3. definition of the initial artefacts' content.
4. set-up of the tool infrastructure
5. definition of the activities and the milestones.

Hint: *Definition of the Process Frame in Comparison to other Approaches*
Two particular approaches that include in this context the customisation of the produced artefacts are the V-Modell and the Requirements Abstraction Model (RAM) [GW06]. The V-Modell addresses with the term (static) *tailoring* the definition of roles, artefact structures and milestones, all steps being tool supported (see also Sect. B.3). RAM, however, describes although being independent of any domain-specific artefact models the elaboration of initial contents at the basis of given inputs.

4.3.2.2.1 Assign Roles

The roles are defined by assigning individual team members of the project to a predefined set of roles. The assignment is performed according to the responsibilities of the roles. All the roles within a project (beyond the primary ones presented in Sect. 3.10) should be included and each team member should be assigned to a role. Note that it is possible to assign the same team member to same roles, like to the roles *business analyst* and *requirements engineer*. A possible project characteristic implying such an assignment is given by the project complexity. For instance, a small team size aiming at a development project with a low duration, without an explicit need of a detailed business specifications could imply to assign same person to mentioned roles.

4.3.2.2.2 Define Artefacts' Structure

The artefact structure is initially customised according to the project scope, the project type and further organisational requirements (see also Sect. 3.5.5). An example for the latter could be the constraint to use the (document) structure proposed by the IEEE Std. 830-1998 [oEEE98].

The customisation of the artefact structure, however, includes:

1. The identification and creation of the necessary artefact types (for example the requirements specification).
2. The (optional) trimming of specific content items of each artefact type, i.e. the deletion of content items that shall not be elaborated at all.
3. The instantiation of the necessary amount of artefacts of a specific type (e.g. two requirements specifications for two systems under consideration).
4. The definition of the order of the content items, i.e. the structure of each specification document and their chapters.

The first step consists in identifying the artefact types as the main deliverables to be produced for an acceptance. The artefacts are then initially created (see also the status model in Chp. 3.3.2).

For each of the chosen artefact types, the content items can be trimmed according to corresponding sets project parameters. Since every content item has its purpose and value, we do not consider any of the items as optional or mandatory. Hence, each eliminated content item implies a risk to the project why trimming should be performed carefully according to pre-defined patterns (see example 4.1). Note that during the project-specific BISA execution strategy, specific content items are also affected in terms of not being produced due to project parameters that are not known at the beginning. The trimming during set-up concerns a planned and especially approved elimination of chosen content items.

However, the next step includes the instantiation of the necessary amount of artefacts of a type. While the business specification includes the descrption of the overall business processes and business services that are in scope of a project, the requirements specification describes the demands towards single systems. Consequently, if facing a project that includes requirements towards more than one information system, one possibility could be to define several requirements specifications (one for each system[5]). In this case, one requirements specification should include the requirements that have more organisational nature like organisational requirements and integrational ones. Besides this "master specification", further specifications should include the description of the requirements for each of the en-

5 It could also be possible to define all requirements within one specification, but depending on the complexity, respectively amount of requirements, and over and above all on how the acceptance of the artefact types is planned this structure is in most cases unmanageable.

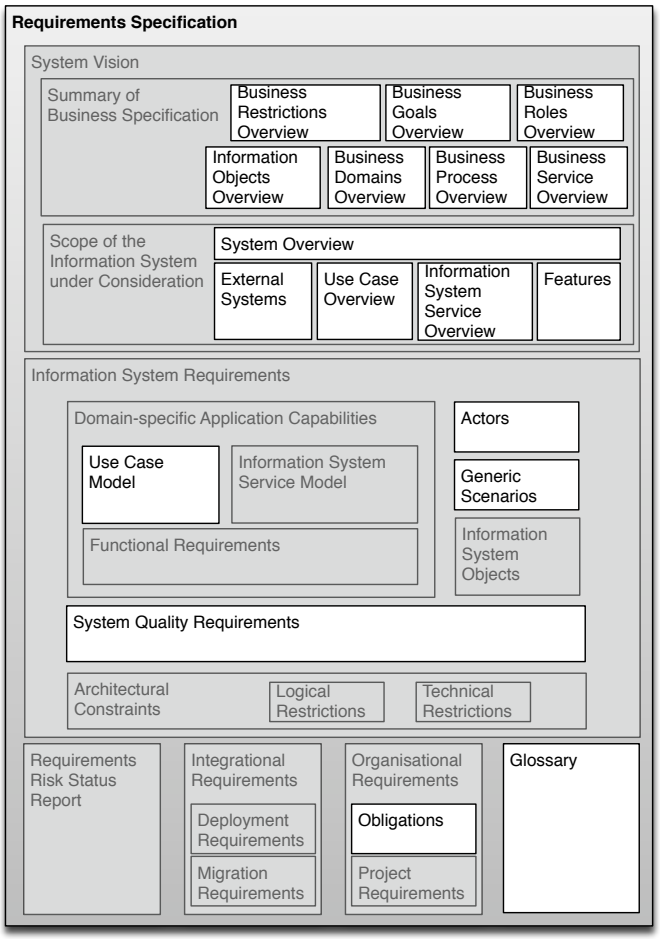
visioned information system, i.e. a system vision and a set of information system requirements.

Finally, the last step within the definition of the artefact structure includes the arrangement of the order of the content items, this means the order of the chapters. Appendix A.1 depicts possible ways of structuring the business specification and the requirements specification.

Example 4.1. *Exemplarily Trimmed Content Items*

An example could be to pre-select subsequently depicted content items of the requirements specification according to a possible project type "lightweight development project" that includes project parameters (and values), like

- *Team size < 10 team members*
- *No external suppliers involved into business BISA and into acceptance tests*
- *Degree of distribution: none, team members situated at customer's company*
- *High degree of communication within team and to customer including scheduled jour fixes*



4.3.2.2.3 Define Initial Artefacts' Content

Once, the artefact structure is defined, each statement within initially given documents has to be classified according to its assertion into specific concept types and finally arranged according to the chosen artefact structure. This step initially prepares the content of the documents for systematically identifying during project execution further needs for action.

This step includes also the initial definition of the visions' content, i.e. the business vision or the system vision. If the business specification does not have to be elaborated, the necessary content (business tasks, business roles, ...) has to be at least sketched within the system vision of the requirements specification (see Sect. 3.5.2.1).

4.3.2.2.4 Set-Up Tool Support

The choice of requirements management (RM) tools strongly depends on aspects like the (expected) amount of requirements, the project duration, and the need of a change management implicating the need of traceability and impact analyses. The more complex the BISA process is expected to be, the more important are the use of RM tools instead of e.g. light-weight document structures like spreadsheets. Often the RM tools are also proposed by customers or involved external parties. However, if using a RM tool, important steps are:

- to ensure the connectivity to the tool being used for design activities since this fundamentally enables traceability.
- to ensure that the used tool(s) ensure during usage the conformance of the models being produced to the concept model.

Because most RM tools are based on a customisable data model, this data model should be adapted to the structure and the content of the concept model defined in appendix A.2.[6]

4.3.2.2.5 Set Activities and Milestones

The final step aims at the definition of activities and milestones. Since the approach is process-agnostig (having no pre-defined activity flow) and the described activities and tasks are coupled to the artefacts, this customisation step concentrates on the definition of milestones. This is done by defining concrete orders and dates for reaching the milestones as an outcome of concrete constraints that arise e.g. from the setting of the bidding procedure. Further going information can be taken from Sect. 3.9.3 introducing the milestones and their possible orders regarding for example waterfall models. Once, the milestones are defined the artefacts are elaborated by using the methods that are coupled to the remaining artefacts (see also the next section).

4.3.3. Approach for Project-Specific BISA Execution Strategy

Once, the documents are initially prepared and the process frame is defined, it is up to

- identify further needs for action and possibilities to act regarding underspecified contents of the artefacts (see Sect. 3.2.3.3)

6 Additional aspects that should be taken into account when choosing a RM tool is the possibility of customising completeness and consistency rules for ensuring the control of conformance and the possibility of exporting documents from the models being produced.

- take adequate decisions on if (and how) to construct missing contents according to influencing project parameters and thereby deciding on the resulting degree of solution-orientation

Compared to the customisation stage described the section before, the execution strategy describes decision taking according to occurrences that arise during project execution and that affect the further elaboration of single contents (like the reliability of the stakeholders). However, as the parameters and the decisions to be taken affect the content of different business domains in a different manner, we first give an overview on the decision stages, before describing the approach for decision taking within the corresponding stages.

The subsequent sections are described based on the assumption that both artefact types the business specification and the requirements specification are to be produced and no content has been trimmed during set-up.

4.3.3.1. Decision Stages

As already introduced in Sect. 3.9.2, we follow the principle that problem statement and problem solving is an iterative approach while each iteration concerning the elaboration of the artefacts' content is performed according to the defined business domains. Hence, the decisions to be performed for the elaboration of the contents affects different business domains in a different manner. While the concepts for the content of one business domain can be completely specified, same concepts can remain in other business domains underspecified. Section 3.2.3.3 gives a definition of the terms relating to underspecified artefacts and the effects on the degree of solution-orientation.

However, according to the different levels of abstraction, we allocate the corresponding stages for making decisions on whether to perform a continuous refinement and / or completion of the corresponding concepts or if to leave the concepts underspecified (incomplete).

Figure 4.8 illustrates this basic refinement path. It illustrates the introduced vertical levels of abstraction and embeds exemplary concepts within each stage (without defining any order for producing the concepts).

In general, the solution-oriented and the problem-oriented path is affected by two essential parameters' impact (scales) that can be formulated as:

1. "What *hinders* me on the completion and the further refinement of the concepts describing workflow?"
2. "What *forces* me to complete and to further refine the concepts describing workflow?"

The term "workflow" describes in this context the elaboration of the concepts that represent the business process logic as well as the concepts that represent the usage of single information systems by means of use cases and further implication of information system's constraints (e.g. of functional requirements). In other words: A solution-oriented description is driven by giving an abstract description of the work that is carried out (using e.g. services) instead of defining behaviour with detailed business task descriptions and use case models[7].

Further concepts are obviously also affected in terms of remaining underspecified since they are directly related to the behaviour description. For instance, the description of the information that is processes (messages described with business

7 It has to be clear that services, for example business services, can be decomposed to a detailed level in which (atomic) services represent single steps that are performed and thereby can be seen also as a workflow description. We represent, however, in the context of this report this refinement as a workflow description with explicit concepts like business tasks and process steps.

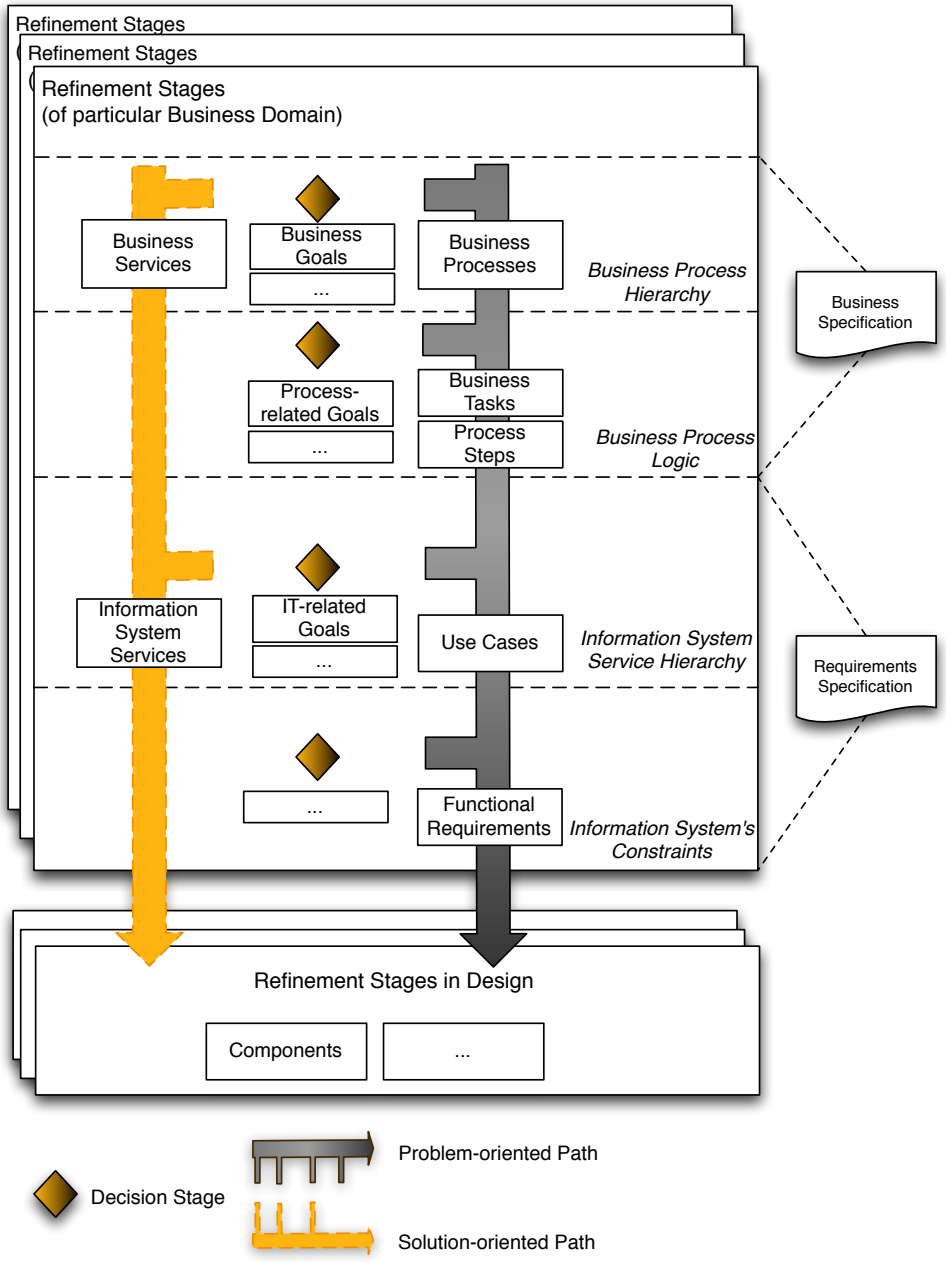


Figure 4.8.: Decision Stages

objects, information objects, ...), participating roles (user groups, actors, ...) or motivating goals. The latter is depicted as an example in Fig. 4.8 in addition to the concepts mentioned above. Hence, a solution-oriented approach can take influence on the abstraction of not only functionality but also for example on the abstraction of quality aspects as these can be described with IT-related goals (like demanding maintainability) or with detailed quantified system quality requirements (see also Sect. 3.2.3.2 describing the principles for refining and decomposing quality aspects).

Depending on the decisions that are taken, the degree of solution-orientation can then be observed in the degree of completion of the overall concepts within the

corresponding level of abstraction, even if focussing in Fig. 4.8 on behaviour descriptions.

4.3.3.2. Approach for Customisation

As described in the fore going section, during the project execution the decisions have to be taken for the elaboration of the content of each business domain according to different levels of abstraction. Figure 4.9 sketches the resulting steps of the approach.

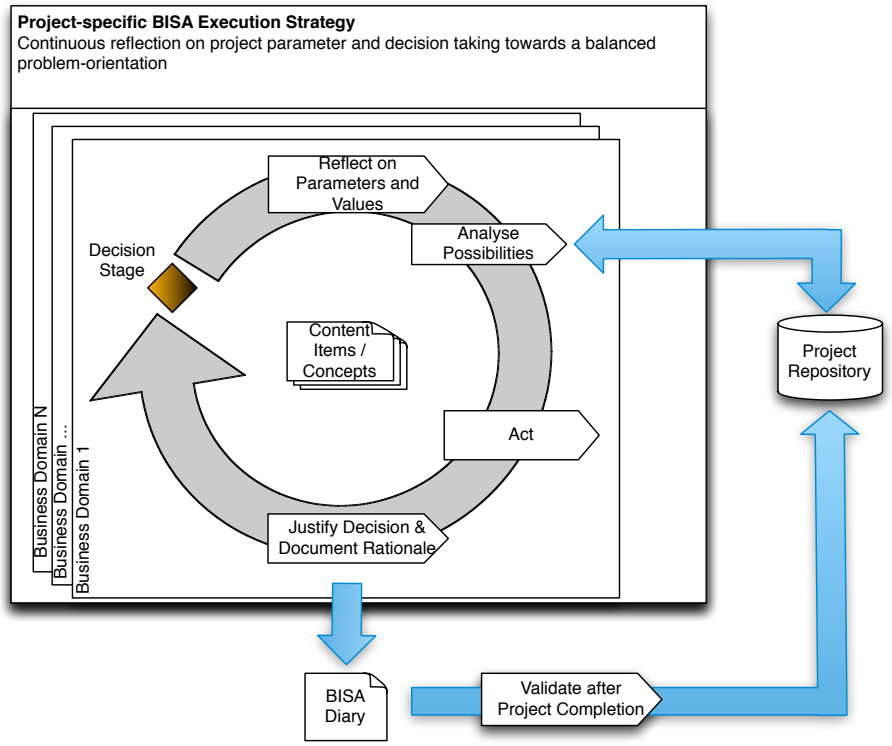


Figure 4.9.: Overview on Execution Strategy

Based on the preliminary defined structure and content of the single artefacts, the remaining underspecified content items, respectively underlying concepts, have to be elaborated. During this elaboration, it is up to reflect on actual project parameters that take effect on the possibilities of constructing the remaining contents, taking adequate decisions and thereby step-by-step constructing the process. These steps, finally, define the possible and necessary degree of solution-orientation within an individual project.

4.3.3.2.1 Reflect on Parameters and Values

For each concept that needs further "action" in terms of creating or completing it, we reflect on the actual individual project situation. We reflect in particular on the *ability to act* and on the *need for action*, i.e. identify the project parameters and their values impacting on the envisioned concepts.

4.3.3.2.2 Analyse Possibilities and Act

The possibilities of whether to elaborate the envisioned concept or not are analysed according to made experiences in other projects. Referring for this purpose to the project repository, we check on same and similar parameters and values that took effect on the elaboration of same concepts in other projects. This is done by comparing the parameters and following the impact scales from the envisioned concepts backwards to the decision that has been taken in further projects.

The own situation and the possible decision to be taken—if to elaborate the envisioned concept or not—is analysed then according to the rating of the decisions in other projects.

4.3.3.2.3 Justify Decisions and Document Rationale

After making the decision regarding the content it is documented as a rationale within the BISA diary, in particular within the project-specific BISA execution strategy. This fundamentally enables reproducible results.

4.3.3.2.4 Validate after Project Completion

For using the made experiences in further projects, it is essential to commit the content of the BISA diary into the project repository. Before committing simply the decisions that have been taken, these have to be validated by rating them and allocating this rating to each decision. One possibility is to estimate the resulting costs. For example by giving a low rating if the decision of leaving certain concept underspecified resulted in a change request due to a missing or wrong requirement.

However, when committing the content of the BISA diary into the project repository, existing entries can be extended with additional project parameters that have not been taken into account yet or modified in their ratings (if the decisions resulted in the own project in a different rating as the ones of other projects).

4.4. Integration by Example: V-Modell XT BISA

In this section we describe the integration of BISA using the V-Modell XT as process model (framework). We discuss the integration approach by a short design description of the V-Modell XT BISA.

4.4.1. Design of the Product Model

As shown in appendix B.1, work products are structered by topics. A top-level BISA artefact is mapped to a work product. Comprised chapters (content items) of an artefact are mapped to recursively structured topics and sub-topics of the V-Modell.

Example 4.2. *Mapping of BISA Artefacts to V-Modell Work Products*

BISA defines the artefact Requirements Specification*, which is mapped to the work product* Requirements Specification*. Content items of the* Requirements Specification*, e.g. the* System Vision*, are mapped to a topic that belongs to the work product.*

The V-Modell requires a role to be responsible for a work product. BISA contains some roles, which are either newly implemented or reused from the reference model. For each work product, which is created from a BISA artefact, the necessary role associations have to be established.

As BISA only defines an abstract process model, the V-Modell requires assignments of work products to so-called decision gates. So for each work product that is created from a BISA artefact, an assignment should be defined if the work product is relevant for the determination of project progress or relevant for releasing a following project stage.

Example 4.3. *Mapping of Decision Gates*

BISA defines the artefact Requirements Specification*, which contains the requirements neccessary to start an implementation. The requirements should have been quality assured to release the implementation stage. So this work product has to be finished to a decision gate, e.g.* System specified*.*

The artefacts to be considered are the

- Business Specification
- Requirements Specification
- Traceability Matrix

We subsequently present the design of the corresponding work products derived from the BISA artefacts. We first introduce the work product structure. Second, we present the integration in the basic structures referring roles and decision gates to complete the static process model parts.

4.4.1.1. Work Product Structure

For the design of the V-Modell work products we use the top-level view of BISA as shown in Sect. 3.3. This view is used to construct the basic structure. Finer grained structures are added in further incremental steps.

4.4.1.1.1 Business Specification

The *Business Specification* is disigned as a so-called *intial* work product type. Initial in that case means, in each project one exemplar of this type has to be created. This ensures that the *Business Specification* is always created.

The *Business Specification* work product is structured by the following topics:

- Business Vision
- Organisation Structure & Business Domains
- Business Goals & Constraints
- Business Capabilities
- Business Information Model
- Business Roles
- Business Demands Analysis
- Glossary

The *Business Specification* is designed as shown in Fig. 4.10. The responsible role is the *Business Analyst*. The activity that creates the *Business Specification* is named *Create Business Specification*. Following the philosophy of the V-Modell, only one activity is assigned to a work product. More detailed definitions of tasks and methods are ommited at first.

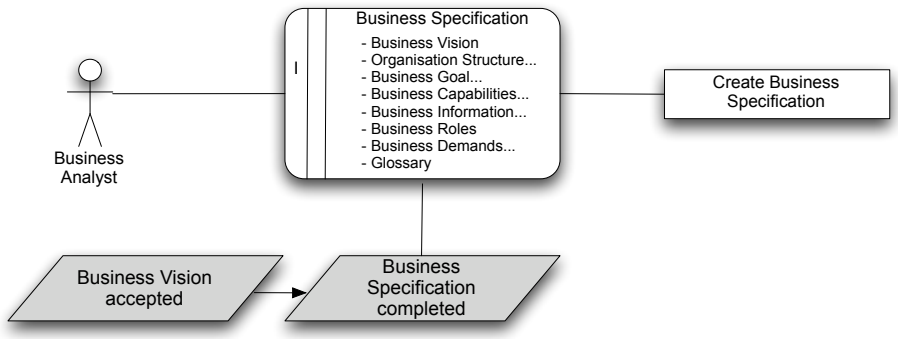


Figure 4.10.: Design of the Business Specification

The *Business Specification* is step-wise developed. To express this, two decision gates are introduced:

- Business Vision accepted
- Business Specification completed

The first decision gate is optional; the second one mandantory (*blocking*). The semantics is as follows: If the the *Business Specification* is developed, the topic *Business Vison* can be indepentantly quality assured. The whole *Business Specification* has to be finished by the decision gate *Business Specification completed* to release the next project stages. At this point the *Business Vision* has also to be completed.

4.4.1.1.2 Requirements Specification

The *Requirements Specification* is also designed as work product type. It is structured by the following topics:

- System Vision
- Information System Requirements
- Integrational Requirements
- Organisational Requirements
- Requirements Risk Status Report
- Glossary

The *Requirements Specification* is designed as shown in Fig. 4.11.

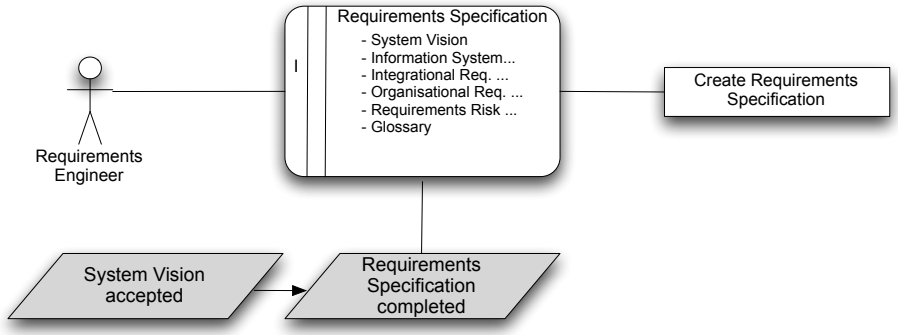


Figure 4.11.: Design of the Requirements Specification

The responsible role is the *Requirements Engineer*. The creation of the work product is described by the activity *Create Requirements Specification* and can also be examined via two decision gates:

- System Vision accapted
- System Specification completed

Analogue to the *Business Specification*-related decision gates, the first decision gate is optional for an additional quality assurance regarding the *System Vision* topic of the *Requirements Specification*. The second decision gate requires a preliminary finished exemplar of the *Requirements Specification* to release further project stages.

In contrast to the *Business Specification*, the *Requirements Specification* must scale for different scenarios of application. If only the analysis and contracting is the scope of the project (customer view), only the topics *System Vision* and *Organisational Requirements* are under consideration. The other topics become relevant only if the project contains development tasks. This is an important note as it influences the model strucutre respecting the tailoring requirements (see Sect. 4.4.3).

4.4.1.2. Embedding Work Products

The newly introduced work products replace the work products from the standard V-Modell. So they have to be embedded into the structures of the whole model to ensure the consistency. In detail, the following integrational aspects have to be considered:

- Some existing work products have to be removed – associations have to be rebuilt to guarantee consistency.
- Generative work product dependencies have to be adjusted.
- Content-related work product dependencies have to be adjusted.

The *Business Specification* replaces[8] the *Requirements Specification* given by the V-Modell. Thus, all generative work product dependencies for that the *Requirements Specification* is a source, have to be adjusted so that the *Business Specification* is now the new source.

The same way the *Business Specification* replaces the *Requirements Specification* (V-Modell), the *Requirements Specification* (BISA) replaces the *Overall System Specification*[9].

Content-related work product dependencies built the content structure regarding consistency and quality assurance between particular work products. Since the *Business Specification* and the *Requirements Specification* replace work products given by the standard, they have at first to be embedded into the dependency structure defined over these standard products. Furthermore, new dependencies have to be taken into account, too.

4.4.2. Design of the Process Model

Having roughly defined the mapping of the BISA artefacts to adequate V-Modell work products, the integration in the process model is the next step. As mentioned above, new decision gate types are introduced to make the stepwise development of the *Business Specification* and the *Requirements Specification* explicit.

8 As the *Business Specification* has a completely different structure and the methodology to create it differs from the standard, no variability operation to rename and restructure the standard work product is performed. The model structures related to the *Business Specification* are created new.

9 Note that the information system requirements describe in BISA demands towards the overall system the same way as described by the overall system specification in the V-Modell (that considers the system also as a black box with no component architecture).

4.4.2.1. BISA Procedure Modules

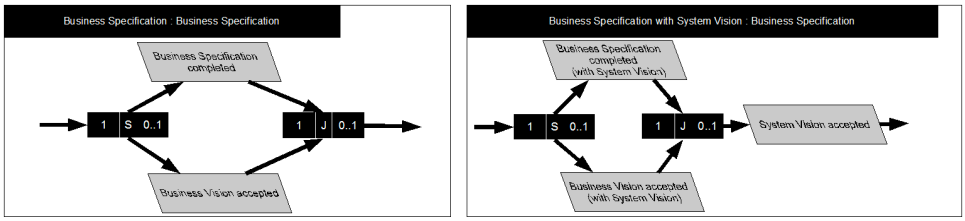


Figure 4.12.: New Procedure Modules for BISA

Fig. 4.12 shows the new procedure modules. The left procedure module describes the (planning) process only for the *Business Specification* (with an optional quality assurance for the *Business Vision*). In the right procedure module, the process description is extended for considering the *System Vision*, too. The processes realise a parallel development of the *Business Specification* and the important topic *Business Vision*. The cardinality states how often a parallel path can be entered. This means, the *Business Specification* has to be developed mandantory (cardinality 1). The explicit creation and quality assurance of the *Business Vision* is optional (cardinality $0 \ldots 1$). The process also defines by the semantics of the *Join*, if the *Business Vision* is inspecteted separately, this quality assurance measure has to be finished (at latest) together with the quality assurance of the *Business Specification*.

If in addition to developing the *Business Specification* a *System Vision* is required, the right procedure module must be chosen for creating the milestone plan. Having quality assured the *Business Specification*, the *System Vision* can be developed.

4.4.2.2. Customer Development Strategy

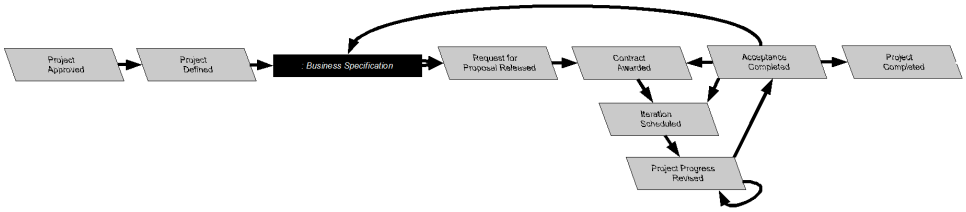


Figure 4.13.: New Processes' Integration into the Development Strategy (Customer)

The new procedure modules are integrated into the customer's development strategy as shown in Fig. 4.13. The "old" decision gate from the standard V-Modell is replaced by a place holder of type *Business Specification*. Depending on the context either the left or the right procedure module form Fig. 4.12 are selected for planning.

4.4.2.3. Customer/Supplier Development Strategy

The same way the customer's development strategy is enhanced by BISA, the development strategy for the project typ *Customer/Supplier* is enhanced. Fig. 4.14 shows the enhanced development strategy.

The "old" decision gate for the requirements is also replaced by the new BISA procedure modules. Differently to the customer's strategy, the *System Vision* is

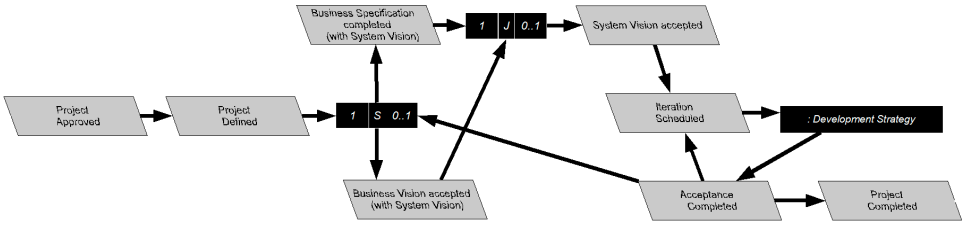


Figure 4.14.: New Processes' Integration into the Development Strategy (Customer/Supplier)

always a part of the development strategy, while the right procedure module from Fig. 4.12 is here not relevant.

As described in Sect. 4.4.1 a new desicion gate type *Requirements Specification completed* has to be introduced. Concrete decision gates of this type are not shown in Fig. 4.14. The reason is that the new decision gate type equals the type *System specified* provided by the V-Modell standard. This standard type will be adopted, but is integrated in the concrete development strategies (sketched in Fig. 4.14 by the *Development Strategy* placeholder).

4.4.2.4. Work Product Association

The new procedure modules have associations to work products defined by BISA. The procedure decision gates *Business Specification completed* and *Business Specification completed (with System vision)* are both of decision gate type *Business Specification completed* and therefore have associations to the *Business Specification*. This means, that this work product has to be in a quality assured state at this decision gate. As mentioned above, the decision gates connect the static process description parts with the dynamice process execution parts. In detail, the modelled development strategy defines when a *Business Specification* has to be developed in a project.

All defined decision gates are, as stated in Sect. 4.4.1, assigned to particular work products or topics, which is reflected by the decision gates' names. The decision gates *Business Specification completed* and *Requirements Specification completed* are "blocking" decision gates.

4.4.3. Design of the Tailoring

In the last sections we have shortly described the design of the work product and the process model. During the design of the process model we also provided detailed design of the new procedure modules. Procedure modules and their associations to project types and project type variants are matter of the *Tailoring* (of the V-Modell), which we describe in this section.

4.4.3.1. Process Modules

In Sect. 4.4.1 we only provided a raw view on the new (central) work products for BISA. To enable the BISA extensions taking part in the tailoring and being able to create a project-specific process, process modules have to be designed.

For the first release of V-Modell BISA two process modules are designed (see Fig. 4.15 and 4.16).

The fist process module contains the central work products *Business Specification* and *Requirements Specification*. Furthermore it contains the work product *Project*

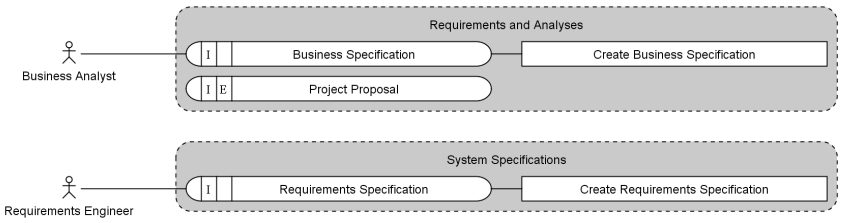


Figure 4.15.: New Process Modules for BISA (Core)

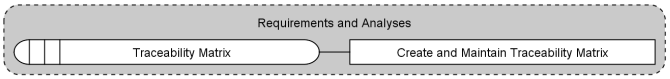


Figure 4.16.: New Process Modules for BISA (Extension)

Propsal. This is required as the new BISA process module replaces its equivalent from the standard V-Modell. The proposal is required to start a project and holds a lot of dependencies to several work products of the V-Modell. Those have to be copied to provide a defined project set-up as defined in the standard.

The second process module contains some extension such as additional, development-relevant topics for the *Requirements Specification* and an addtional *Traceability Matrix*. Depending on the project context, only the core-module or both are referred in the tailoring.

4.4.3.2. Project Types and Type Variants

Tailoring establishes content and process structures. To connect both, project types are used. BISA supports two project types from the standard model. Furthermore each project type requires at least one project type variant, which adds concrete procedure modules to the content-frame. The concrete constellation in the V-Modell BISA extension is as follows:

- Project Type: System Development Project (Customer)
  - Project Type Variant: Project (Customer) with One Supplier
- Project Type: System Development Project (Customer/Supplier)
  - Project Type Variant: Project (Customer/Supplier) Including SW-Development, Enhancement or Migration

The design is close to the standard components. Only the scope of the project type variant "Project (Customer/Supplier) Including SW-Development, Enhancement or Migration" is restricted to software development. So some of the project characteristics of the standard V-Modell are left out.

4.4.3.3. Project Characteristics

As project types and project type variants define the mandantory parts of the process, project characteristics define the optional parts. The V-Modell BISA extension contains the following subset of project characteristics:

- Life Cycle Cost Management
- Off-the-Shelf Products

- User Interface
- Subcontract
- Legacy System
- Prototype Development

Those project characteristics have relations to the requirements engineering focus of BISA. Other charateristics are left out or are integrated in the project types or type variants, e.g. *Subject of the Project*, which is not neccessary as BISA is focussed to software development. The components of the process that are relevant to software development are not referred using a characteristic but are referred directly by the project type variant.

4.4.4. Realisation

The BISA extension are realised using the V-Modell XT R/O mechanism. This concept divedes the whol process model into a *R*eference model and an *O*rgansational extension model. The latter contains all enhancements and all changes related to the standard process model.

For BISA the extension model contains:

- the new process modules
- the new procedure modules
- the re-designed project types and project type variants

Furthermore the extension model contains a set of change (variability) operations used to customise the reference model. In detail, operations for deleting content, changing structure and replacing texts are used.

5. Discussion

In this report we provided an artefact-based reference approach for the application domain of business information systems. Our goal was to tackle the problem of variability in designing a process and the unawareness of producing underspecified (incomplete and inconsistent) specification documents as part of a solution-oriented RE process. We aimed therefore at an approach that:

1. serves as a reference model and supports awareness of producing precise results that are conformant to the basic concepts of the application domain
2. can be integrated into organisational environments, i.e. integrated into existing process frameworks
3. can be customised to volatile project environments while incorporating experiences and the organisational culture and finally achieving transparency with reproducible decisions and consistent results

We developed at this basis a customisable reference model that supports a systematic business information systems' analysis. The reference model enables seamless modelling of precise and traceable results and thereby continuity within the process by making use of the artefact-based paradigm. This paradigm also enables due to its modular structure a systematic customisation to organisational environments and to project environments.

In a first step, however, we analysed in chapter 1.5 existing domain-specific standards for describing information systems in order to show to what extent RE aspects should be covered within the application domain. Taking the steering principles of RE that aim at IT business alignment we defined under this perspective for our context the discipline business information systems' analysis.

In chapter 2 we introduced then a meta model that defines structure and semantics of the artefact-based paradigm since the term "artefact" is differently used in same and similar contexts. In chapter 3 we proposed at the basis of the meta model and the principles of business information systems' analysis the artefact-based core model. This core model includes:

1. an artefact abstraction model defining horizontal and vertical abstraction (refinement hierarchies) what in turn enables precise decisions on when artefacts are underspecified and the process therefore solution-oriented
2. an artefact model as an abstraction of structure of specification documents and the included concepts reflecting the elements of different description techniques (and related to the artefact abstraction model).
3. a process model that defines milestones and methods for elaborating the artefacts, directly coupled to artefacts.
4. a role model that defines responsibilities for the proposed artefacts

Based on this artefact-based reference model, we defined in chapter 4 a multi-staged customisation approach. The approach introduced the mechanisms for a systematic and transparent customisation of the reference approach to (1) organisational environments and to (2) individual project environments.

Regarding the customisation to project environments we showed how project parameters that characterise volatile project environments affect the construction of the different parts of the artefact-based core model (artefacts, milestones, roles,

...). At this basis we illustrated similarly as done in decision support systems how to use a project repository that incorporates different project parameters and their (experienced) effects onto the core model. Using this repository we showed how to systematically elaborate according to the influencing parameters the artefact model. The approach for this elaboration is performed over two stages. Firstly, by defining a process frame including artefacts structures, milestones and initial contents. Secondly, by deciding during project execution what contents to leave underspecified and what contents should underlie further refinement, i.e. by deciding on the degree of solution-orientation. In particular, we coupled exemplary project parameters like the availability of end users (gained from an exemplary process integration) to selected artefacts' contents and showed with the customisation approach on what parameters to reflect for each artefact, what decisions to take regarding their construction (or if leaving them underspecified) and finally how to document these decisions within the BISA diary. The BISA diary finally serves for ascribing the taken decisions to the project repository and finally for achieving transparency with reproducible results. However, since the project parameters and the decisions are coupled to chosen elements of the artefact model, the customisation approach still respects the interdependencies between single artefacts, and thereby ensures consistency during customisation, thus, continuity in the development process.

Finally, regarding the customisation to organisational environments, we illustrated the process integration in chapter 4 with the V-Modell XT, a German standard for software and systems development projects. We chose relevant project types that fit the application domain of BISA. Based on these project types, we extended corresponding product types of the V-Modell and embedded them into the process structure.

In conclusion, we provided an approach that is deeply integrated into the development life cycle and prepared its application in real life projects. We laid the foundation for a systematic and integrated business information systems' analysis that supports in individual project environments consistency and completeness of reproducible results, continuity within the process and variability in the process definitions.

5.1. Validity of the Approach

The validity of the approach is threatened by several aspects. We refer, in particular, to the correctness of the concept model, to the applicability of the process model and finally to the validity of the customisation approach.

5.1.1. Correctness of the Artefact Model

We do not claim for completeness of the artefact model and its use as a silver bullet since there is no universal way of modelling requirements[1]. Still, we claim for the model's correctness. The reason is that the concept model has been developed by inferring least common denominators of elements and relations found in different description techniques. For instance, the concepts of business processes are defined according to the underlying principles of different techniques and notions like BPMN and EPK, but also approaches with an underlying mathematical theory like the approach of Thurner [Thu04].

1 Note that the description of a component architecture as part of the artefact model (as part of system specification documents) is intentionally left often since there exist established standards that can be used for this purpose.

Hence, we defined a model that abstracts from established and correct approaches enabling the precise definition of specification contents. In addition, the application of the different concepts and their relations are evaluated according to documented and lived experiences in real life projects of the company Capgemini sd&m. Furthermore, it is integrated into the organisational process model of the company and trained (including its tool-supported appliance) as the standard reference approach for business information systems' analysis. Regarding the artefact structure, it is defined according to the concept model and thereby is it also correct. It provides the possibility of customising it to product-based standards like the *IEEE Recommended Practice for Software Requirements Specifications* [oEEE98] (see also Chp. 4). One customisation has been showed by integrating the introduced artefact model into the product model of the V-Modell XT.

5.1.2. Applicability of the Process Model

The process model does intentionally not define a strict order for producing the artefacts. In addition it does intentionally not define concrete methods that are restricted to a chosen syntax. Both aspects guarantee flexibility in the process definitions during customisation to organisational environments and to project environments. However, the applicability is given since we integrated the reference approach into the organisational standard process descriptions of the company Capgemini sd&m (as mentioned before). During this integration, the process is adapted to the elements defined by the *Rational Unified Process* (RUP) [JBR99]. Furthermore, a chain of concrete methods using different languages that are provided for example by the UML is implemented. Further information can be taken from [MFsA09].

5.1.3. Validity of the Customisation Approach

Regarding the validity of the customisation approach we distinguish between the customisation to organisational environments (process integration) and the customisation to project environments.

Validity of the Process Integration. In addition to the mentioned integration into RUP, we showed in this report a detailed customisation to organisational environments by using the process framework V-Modell XT. Still, the process integration has limitations since a mandatory prerequisite for this integration is that the targeted process framework must exhibit a modular structure that is ideally based on a meta model (as it is given by the V-Modell). It should at least offer a precise description of the single elements of the framework and thereby allow the extraction of such meta model (see also Sect. 4.2.2). A consequence of this restriction is that an integration of the artefact-based core model cannot be clearly integrated into informally described approaches as it is the case with *Scrum*, even if some aspects of "agile approaches" like the lightweight documentation are explicitly supported (see also the trimming procedure in Sect. 4.3).

Validity of the Customisation to Project Environments. Regarding the customisation stages at project level we ensure also validity, although we exclusively provide the basic concepts. One important goal during development of the introduced customisation approach was to ensure the validity of the introduced concepts by clearly separating individual (experience-based) aspects of decision taking and the commonly applicable mechanisms. We restricted for this purpose individual aspects of companies to be exclusively reflected by the values of project parameters

and the made decisions (kept as impacts and impact scales in a structured project repository), not by the parameters themselves that still remain valid. For example, the parameter that reflects the availability of the end users takes effect on the construction of a use case, but how the effect is finally defined (whether to construct exactly what parts) depends on individual aspects such as experiences, political and strategic interests of the team members. Still, when setting up the used project repository within a company, it needs for a training phase. The validity of the parameters and their values respectively the rating of the made decisions, however, increases with each project that documents its experiences (within a certain learning curve).

Hence, we ensured the validity of the customisation to project environments and prepared its in volatile project environments.

Finally, we did intentionally not incorporate the decision of a concrete syntax into the customisation approach at project environments. As the explicit definition of syntactical dependencies between single artefacts would make the artefact model highly complex[2], the decision on what syntax to choose during customisation would obstruct the applicability of the customisation approach.

5.2. Future Work

We are currently enriching the artefact model with conformance constraints by mathematically defining completeness and consistency conditions. The purpose is to enable tool-supported verification in order to control the conformance of the models being produced to the artefact model.

At this basis we are currently preparing an evaluation of the results of this report (the integration of the BISA approach into the V-Modell) by performing an action research study in a real life project with a duration of one year.

Finally, we are analysing in a survey a set of project parameters and their effects (including additional European companies). We aim at the implementation of a prototypical project repository that is ready for use. The survey is necessary because the project repository would otherwise only include impact scales and values of one particular company (Capgemini sd&m). Within this implementation we also develop a meta model for project characterisation with concrete impact scales that take effect onto the roles and the milestones (going beyond the effects of project parameters on the construction of artefacts).

2 "How does a concrete syntax of an artefact *A* match with the syntax of an artefact *B*?"

A. Comprehensive Artefact Model

A.1. Artefact Structure

A.1.1. Business Specification

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

A.1.2. Requirements Specification

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 Maintainbility Requirements
    - 3.4.3 ...

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

A.2. Concept Model

We subsequently illustrate the comprehensive concept model for business information systems' analysis. We first depict the concept model of the business specification, afterwards the one of the requirements specification and finally conclude with illustrating the dependencies between both. The latter illustrates in the nearer sense the principle of IT business alignment (going beyond the concepts of the system vision).

A.2.1. Business Specification

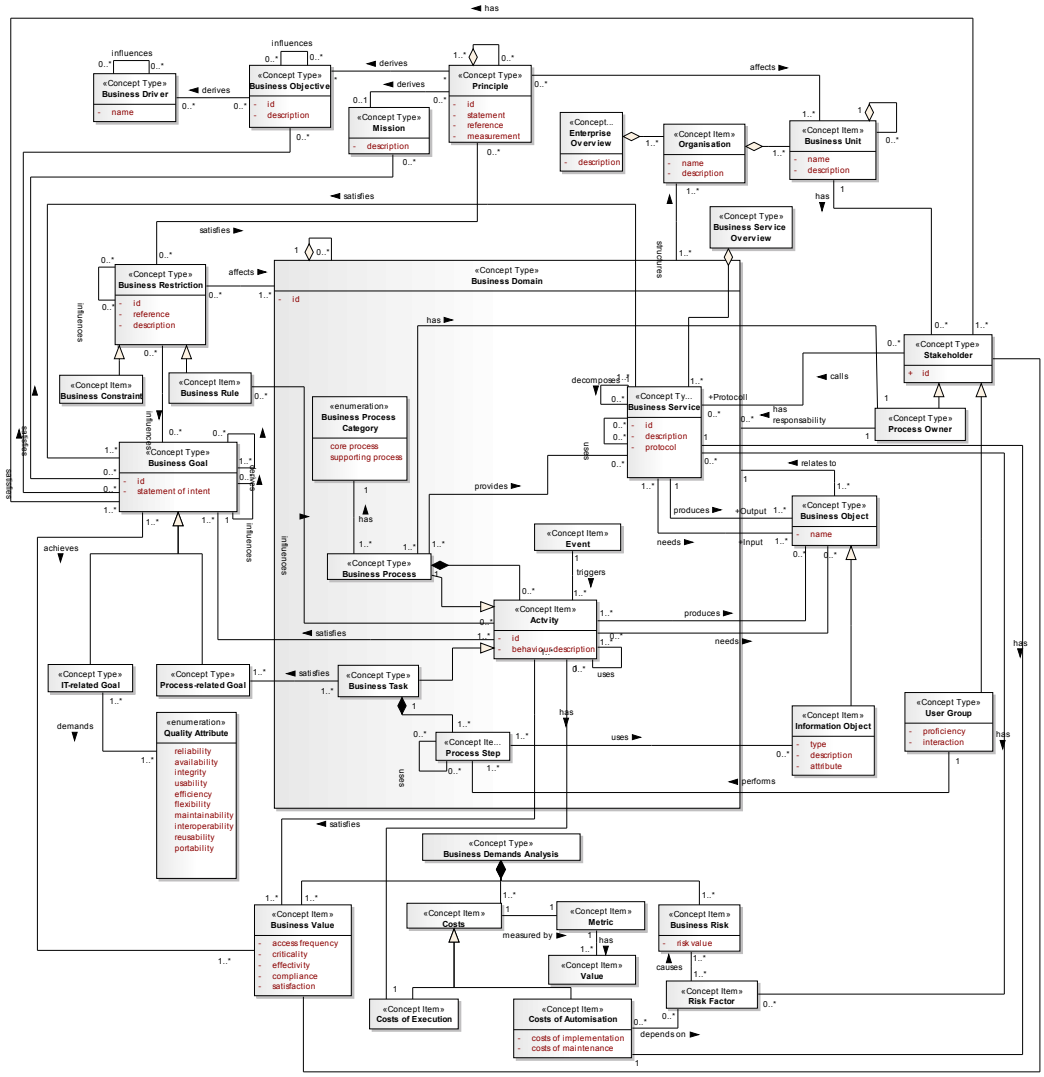


Figure A.1.: Concept Model of the Business Specification

A.2.2. Requirements Specification

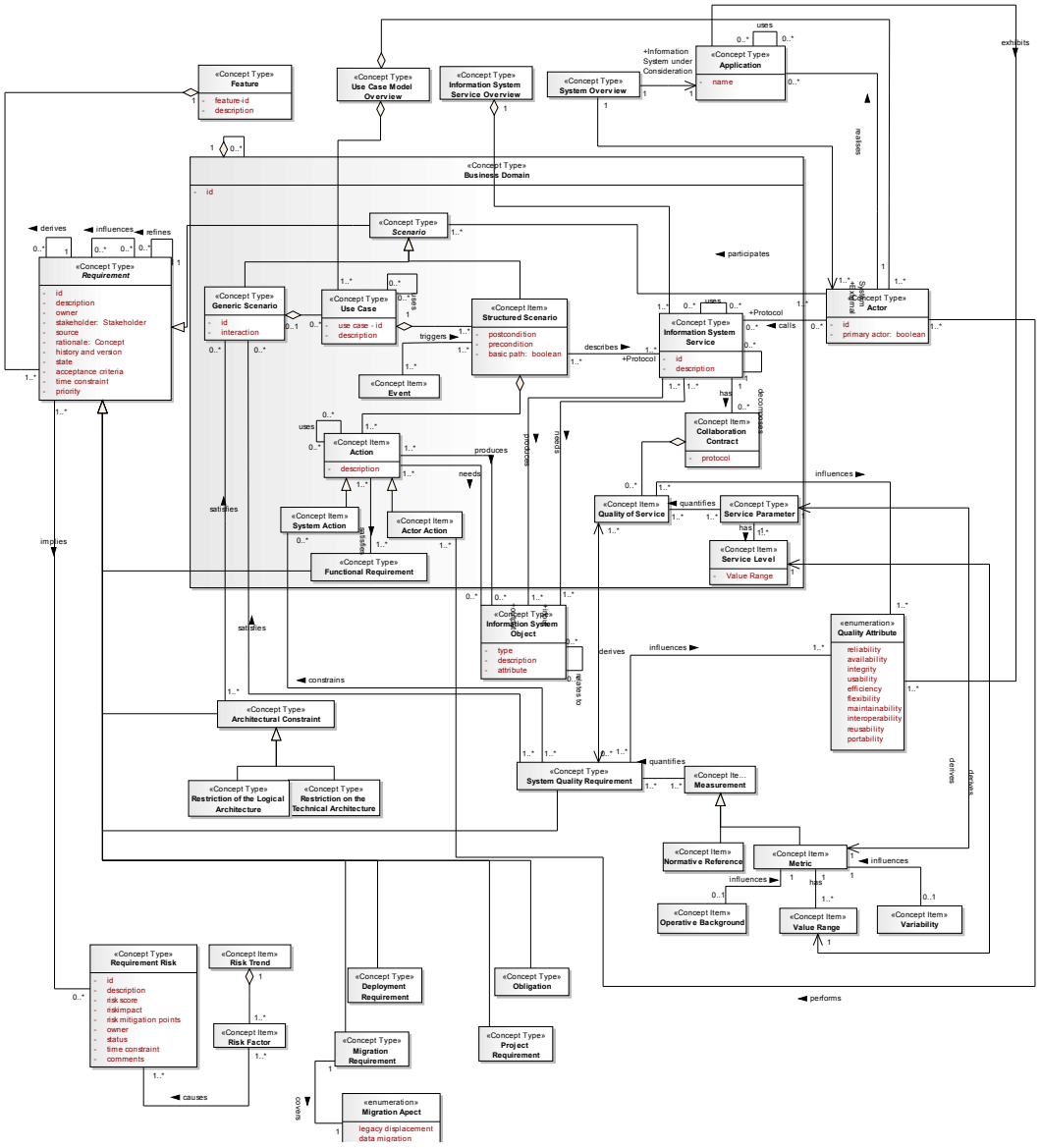


Figure A.2.: Concept Model of the Requirements Specification

A.2.3. Transition from Business to Requirements Specification

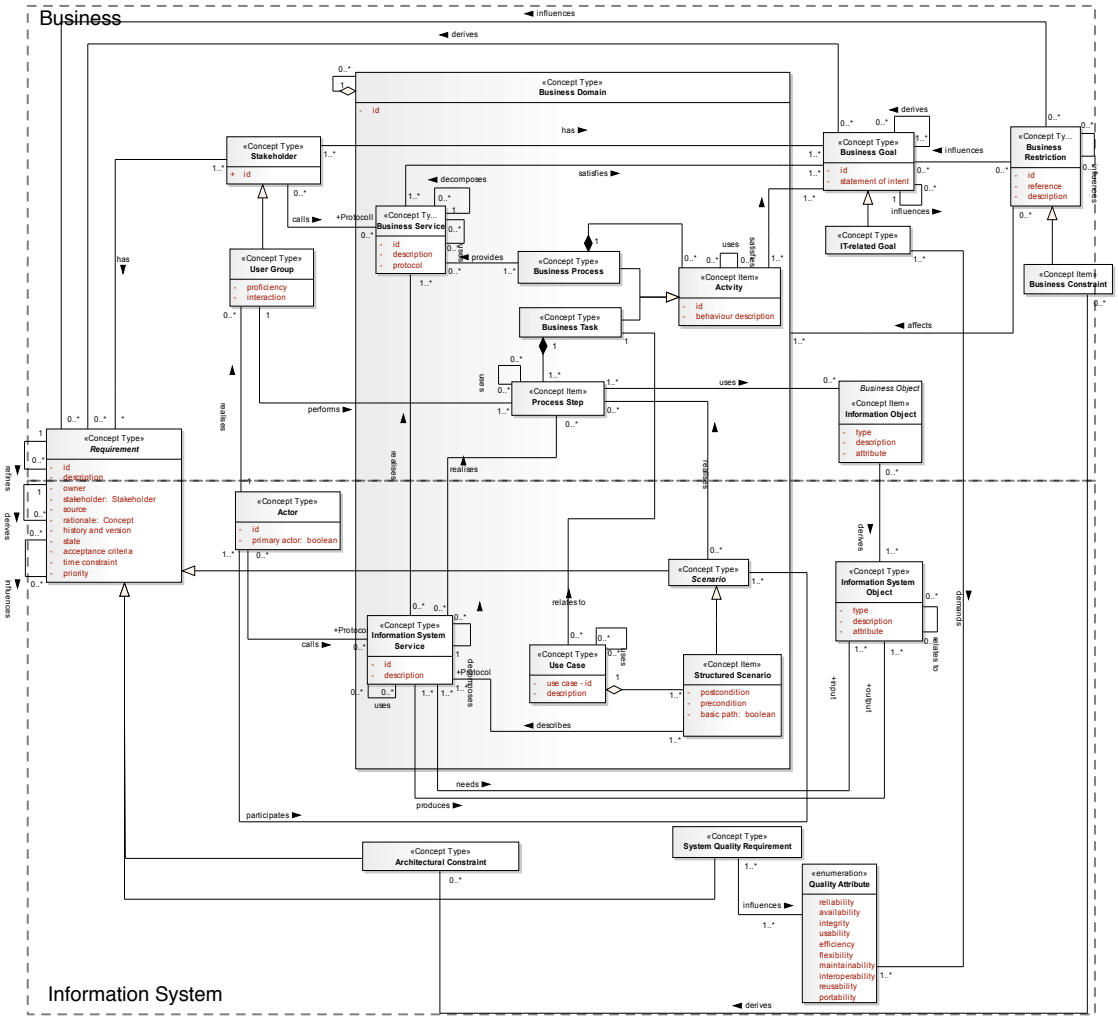


Figure A.3.: Conceptual Dependencies between the Business and the Requirements Specification

A.3. Requirements Shell

Figure A.4 depicts a requirements shell for documenting (initially stated) requirements in an informal way, similarly as known from the Volere requirements specification templates [RR07]. The shell is defined according to the artefact model of the requirements specification and facilitates the initial documentation of requirements for example within workshops, before adjusting the description according to the concept model. The original statements (the shells) serve then as a rational for the content within the requirements specification.

Requirements Shell		
ID	*What is the unique identifier of the requirement?*	
(Original) Description	*What is the requirement's description?* *(→ Can be transferred in dependency to the assertion into corresponding requirements types, respectively concepts such as structured scenarios or system quality requirements)*	
Owner	*Who is responsible for resolving this requirement?*	
Stakeholder	*Who is the customer-side responsible individual or organisation that demands this requirement?*	
Source	*Do there exist supplementary sources for the requirement, such as a:* • *Document / Specification* • *Email* • *Protocol of a phone call or a workshop*	
Rational	*What artefacts of the business specification are the rational for the requirement?*	
Influences on	*What further requirements are influenced by this requirement? Are they conflicted? Do they support each other?*	
Derived from	*From what requirement is this one derived?*	
History & Version	*Is this requirement new? Was it changed? When was it changed? What changes have been made? What is the resulting new version?*	
Business Value	*What business value 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 to Business Rule	*Does the requirement result from a not negotiable business rule (such as a law)?*
State	*What is the actual state of the requirement?*	
Acceptance Criteria	*When is the requirement realised achieving the expectations of the stakeholder?*	
Time Constraint (optional)	*Until when (or within what release) does the requirement have to be realised?*	
Business Domain (optional)	*To what business domain is the requirement related?*	

Figure A.4.: Shell for Capturing Requirements

B. V-Modell XT Meta Model

All concepts of the V-Modell are precisely defined by a formal meta model [TK09]. Thus, syntax and semantics are clearly defined, i.e. for each work product exactly one role is responsible. In the following paragraphs we give a short impression of the relevant parts of the meta model for this report. As we focus on artefact-orientation, we only discuss the static work product-related parts of the meta model.

B.1. Static View: Work Products

The meta model is divided into specialised sub-models providing views. The first view to be considered is the view of work products, which is part of the so-called static sub-model.

The static sub-model contains all elements that are used to define, structure and describe the process. It contains work products, activites, roles and so on. This sub-model is used for generating the process documentation using the content description of the particular model elements.

B.1.1. Work Products

Work products define the structure of the (potential) results of a project – also called *result structure*. We focus on the work product view in the following.

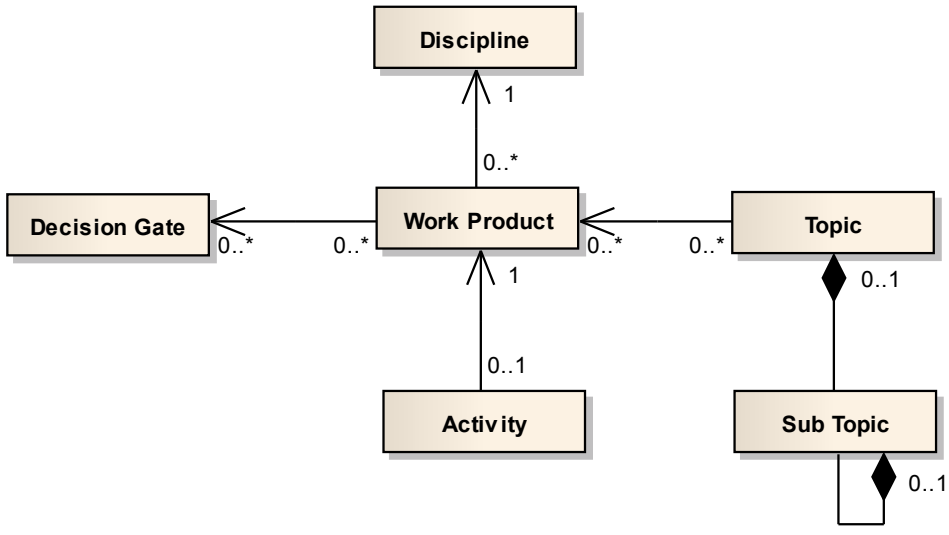


Figure B.1.: Meta Model Aspect: Static View on Work Products

Work products (Figure B.1) are the basic concept. They are structured by *topics* covering certain contents that should be addressed by a particular work product. In other words: topics define what contents a work product shall contain. To provide

fine grained modelling of work products, topics can also be structured by so-called *sub topics*, which can recursively contain further sub topics. The V-Modell provides containers that combine work products that are assigned to similar project stages/phases. Those containers are called *disciplines*. Disciplines also contain *activites* that are used – in the V-Modell-context – to describe the finalisation of work products. Compared to project planning, V-Modell-activities are more comparable to work packages than to (elementary) tasks.

As no activity flows can be expressed, because of the unconnected activities, the V-Modell provides another concept to define an order for products' creation. Work products are assigned to *decision gates*, which become milestones in a project plan. The semantics is: If a decision gate (milestone) shall be reached, the set of assigned work products have to be finished, including successfull quality-assurance. Decision gates are the interface to the *dynamic* sub-model that defines an order of decision gates. So the order of work product creation is given by the order of decision gates combined into the *project execution strategy* (cf. section B.2).

B.1.2. Work Product Dependencies

Each work product is embedded in a system of dependencies. The meta model provides different dependency types that support the expression of

- work product creation
- relation to work products' contents
- structure of complex work products (respecting their comprised sub-work products)
- necessary changes of the process depending on the work products' contents

Work product dependencies are used to model systems of products using associations as shown in Figure B.2. For this report we only need the *content-related* and the *creational* work product dependencies.

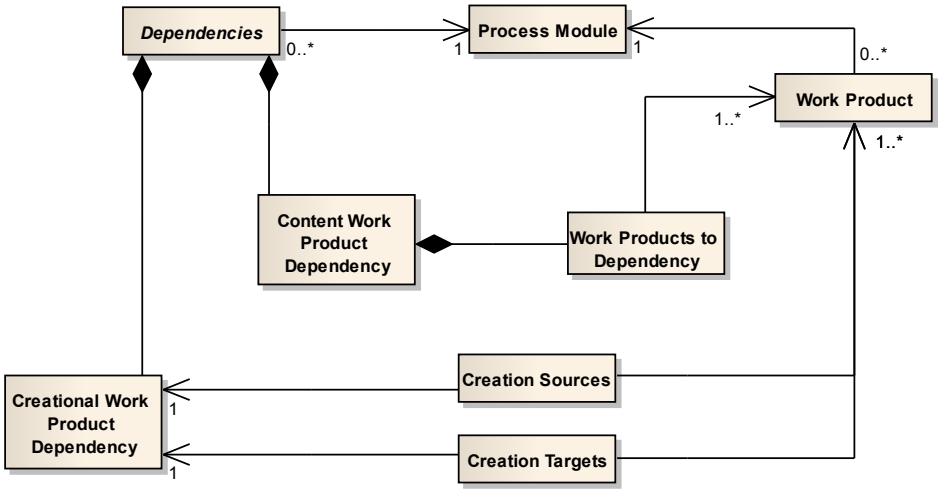


Figure B.2.: Meta Model Aspect: View on Work Product Dependencies

Creational dependencies are used to express *why* a particular work product has to be created in a project. The sources of the directed dependency are the so-called *initial* work products. Those products regulate what other work products have to be created (including shape, intervals and so on). The targets of that dependency

type are the products, which creation is regulated. To give an example, the project plan defines when a status report has to be created.

Content-related dependencies connect work products with each other expressing that contents of one work product depend on the contents of another one. An example is the dependency between the system under development and the system specification.

B.1.3. Relations

While work product dependencies connect only work products, other process elements such as roles, decision gates and so on have to be taken into account, too. To structure and connect the different process elements such as work products and roles, typed and directed *relations* (associations) are used.

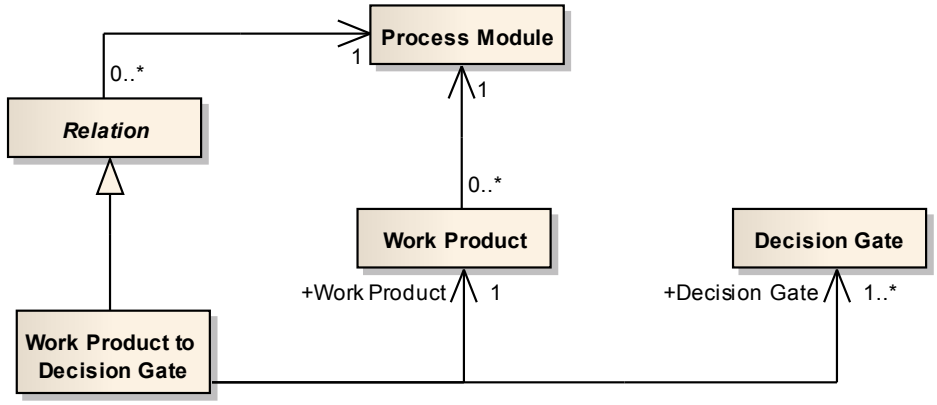


Figure B.3.: Meta Model Aspect: View on Work Product Relations

Figure B.3 shows the relation between work products and decision gates, as described above. The idom is the same for all relation types in the meta model. There exsits a relation type that connects several types of process elements, e.g. work products and decision gates or work products and roles.

Dependencies and realtions are used in the later presented proof of concept. They are used to built the structure of the BISA artefact types and their contents.

B.2. Dynamic View: Project Execution Strategies

The static parts of the meta model define the result and documentation structure of the process. Thus, the V-Modell is artefact-based, no activity flows are defined. To provide a process organisation or a *process structure*, project execution strategies are used. They state what decision gates (milestones) have to be reached in which order. Since all work products are assigned to milestones, the dates for finalisation are defined. So the result structure and the process structure are connected to each other.

The underlying concept is the *procedure module* that defines a partial process, respectively a subset of decision gates in an given order. In a particular project, several procedure modules are used to build a project execution strategy. The concrete selection of relevant modules is given by a concrete *tailoring profile*, which defines the process parts relevant for a concrete project.

B.3. Tailoring View

Tailoring in the V-Modell means an automatic, tool-supported customisation process to adopt the whole process model to the project's specific requirements. During the tailoring, so-called project types and project type variants have to be selected to roughly sketch the project setting. A more detailed customisation is in the following provided by a set of context-specific *project attributes*, which allow the user to select additional options, e.g. to embed sub-contracting parties into a project. In this section we only give a brief overview over tailoring. A detailled description can be found in [TK09].

B.3.1. Process Modules

Process modules are the content-containing components relevant for the tailoring. A project-specific V-Modell instance is a consistent selection of process modules. Since process modules contain all static process elements (cf. section B.1), the result structure for a particular project is given by that selection.

B.3.2. Procedure Modules

Processes and sub-processes, relevant for the project, are defined by the procedure modules. Same as for process modules, a set of procedure modules is selected during the tailoring. All selected modules are used to derive the project-specific project execution strategy.

B.3.3. Project Tailoring Profile

The selection of the different modules builds the project-specific process model. The relations and dependencies are also defined by the meta model. The relevant concepts for the tailoring are the *project types*, which roghly describe the frame for a particular project (e.g. customer or contractor). Project types are supplemented by *project type variants* that add specific process and concrete procedure modules. Project type variants provide an automatically computed project execution strategy for the project. Finally project attributes provide capabilities for fine grained customisation as described above.

The meta model also defines a stepwise tailoring process, which is implemented by a tool. At first a project type has to be choosen. Secondly, one of the context-specific provided project type variants has to be selected and finally – and optional – project atrribute values can be assigned. The meta model structure defines that selection of a project type is a precondition to select a project type variant and so.

The resulting tailored process model is a subset of all contents provided by the whole standard process model. The results can be exported (transformed) into several formats, e.g. PDF oder DOC for process documentation or work product templates.

C. Glossary

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. We distinguish between primary actors that provide and use information and secondary actors that only use information from the system as an input.

Architectural Constraint Architectural constraints are statements that restrict the information system's solution in its architecture respecting its logical and / or physical layers. They demand or exclude specific solutions. See also "restrictions on the logical architecture" and "restrictions on the technical architecture".

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 (or task) describes the sequence of steps that is performed in order to make use of a concept.

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 Domain A business domain groups a set of logically related business processes and their sub-processes, thereby the business services that are provided by the business processes. It defines a major area of the business that is in scope of the architecture. A business domain can be decomposed, but relates in any case to exactly one process owner and its business processes.

Business Driver Business drivers are internal and external factors, including individuals, knowledge and conditions that initiate and support the design of business activities.

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. They are characterised by (1) a boundary, respectively scope, (2) a (functional) behaviour that supports business process logic 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 in this report are *Information Systems (IS)* and *IT Systems*.

Business Information System' Analysis Business Information Systems' Analysis (BISA) is a discipline that aims at describing the problem space as comprehensively as possible. It comprises activities for an iterative and systematic approach of defining a (1) business specification, reflecting the essential needs of all relevant stakeholders towards the current and the future state of the business and (2) requirements specification, reflecting the essential 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 represents a unique object of reality (the "business world"). 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 Business objectives are statements of intent that define planned outcomes to an organisation.

Business Process A business process is a cross-functional sequence of activities across multiple positions in order to achieve a planned work result in an organisation and that contributes to at least one business goal. It is administrated by a process owner and serves to directly or indirectly produce a product or provide a business service for a customer or a market. The activities are triggered by an event and may use other activities to transform (produce, read and modify) one or more business objects and information objects. Business processes can be hierarchically decomposed until reaching the granularity of business tasks.

Business Process Category 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.

Business Rule A business rule is an actionable directive that defines or constrains 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 Service A business service defines a business capability provided by the execution of one or more business processes satisfying at least one business goal. It can be hierarchically decomposed and maps an input (stimulus) to an output (response).

Business Specification A business specification formulates all goals, capabilities, restrictions and conditions that affect the business of a customer's organisation and further information that describe the current and future state of the like.

Business Task Business tasks are partial units that are encompassed by a business process. Each business task consists of an ordered sequence of atomic process steps, while these in turn are (1) performed by exactly one user group aimed at achieving at least one process-related goal (2) can be supported by an information system.

Business Unit A business unit is a recognised part of an organisation under management of at least one stakeholder (see also "organisation").

Business Vision A business vision introduces into the actual problems and defines the resulting scope of the business that shall underlie a change as part of

a project. The business vision steers customer-side decision-makers 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 finally resulting (re-) structures of envisioned organisations.

Conformance A model being produced at the project level is conformant to the (referenced) concept model, if the concept model is an abstraction of the more concrete model. Conformance of a model at the project level to the concept model is (in the context of this report) related to completeness and to consistency. Completeness requires that selected elements of the model considered conformant are present, while consistency requires that selected relationships must not be violated.

Deployment Requirement Deployment requirements describe 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.

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.

External System An external system is a system including one or more applications that directly or indirectly interacts with the envisioned system in order to realise the workflow 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 that fulfils this feature.

Functional Requirement Functional requirements define the demanded external behaviour of an application that is given by a stimulus, followed by an expected response, as part of a reaction to this stimulus, independant from the underlying realisation of this response. Delimited from functional requirements are the characteristics and conditions, which are related to the expected response (see also system quality requirement).

Generic Scenario A generic scenario is an abstract description of an interaction between an actor and the system or its environment with the purpose of (1) describing interactions / behaviour, that has to be prevented (by the means of attack scenarios), (2) describing interactions / behaviour, that directly has to be supported without any relation to the business process logic (e.g. by describing maintenance tasks that in general address the internal quality of an application), (3) structuring, ordering and grouping use cases (emphasising what users have to perform before executing a use case or between executing several use cases

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.

Information System Requirement Information system requirements are requirements that exclusively describe specific demanded functionalities, properties and conditions of the application, its architecture and its environment from a customer's perspective.

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 of updated.

Information System Service An information system service defines a piece of system's functionality to support the realisation of business processes and / or

business services. It can be hierarchically decomposed and maps an input (stimulus) to an output (response). Synonyms: Application Service, IT Service.

IT Information Technology, including aspects of business information systems, their (logical and technical) architecture and finally applications (software).

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.

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 hierarchically structured business units with clearly defined boundaries.

Principle A principle is a statement of belief, approach or intent which serves as a basis for directing the formulation of architectures.

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 A project is performed over a chosen time frame and aims at achieving an objective and a project scope. A project has a specific project type that represents a selection of a set of project parameters with assigned values affecting processes, roles and artefacts' structure and content.

Project Characteristic A project characteristic is a taxonomy of parameter categories and parameters that affect concrete elements of the development process. A parameter category describes a major area of concern that is decomposed into a set of related project parameters. A project parameter is a measurable or at least assessable circumstance that has an impact onto the elements of the process model, the role model and / or the artefact model. Each project parameter is measured or assessed using a metric and a corresponding value range.

Project Requirement Project requirements are constraints towards the content and / or structure of selected artefract types and the process model, i.e. affecting the process design.

Quality Attribute A quality attribute reflects a composition of perceptible characteristics that are exhibited by an application and its environment due to its properties. Quality attributes are hierarchically organised. An example is "Efficiency" that can be decomposed into "time behaviour" and "resource utilisation".

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. Synonyms: Service Level Agreement.

Requirement A requirement is a demanded property of an information system or a process and environment related to the system's development and integration, both to be accomplished by a development project.

Requirements Risk A requirements risk is a defected requirement that causes potential problems to the development activities and to the final software product quality due to specific risk factors.

Requirements Specification A requirements specification encompasses all demanded properties of a system and / or a development process that shall be accomplished by a project in order to satisfy the demands stated in the business specification.

Restriction on the Logical Architecture Restrictions on the logical architecture restrict the of the solution's architecture logically in its structure.

Restriction on the Technical Architecture Restrictions on the technical architecture restrict the architecture of the solution in the used technologies and used hardware.

Risk Factor A risk factor is the root cause for the requirements risk. One risk factor can be the root cause for several requirements risks (defected requirements).

Risk Factor Trend A risk factor trend calculates the total amount of risk factors that are the root cause for the defects.

Risk Trend A risk trend calculates the total amount of requirements that are affected due to the same defects.

Solution / Problem - Orientation Solution-orientation is the approach in which artefacts remain underspecified. They are directly distorted by solution ideas of following constructive development phases. The concepts that describe the problem space remain incomplete at the level of the business process hierarchy and / or the IT system hierarchy. Problem-orientation is the approach of a continuous approval process with a customer in which the artefacts are analysed, refined and quantified in consistency and compliance to the growing logical architecture.

Stakeholder A Stakeholder describes a unique individual, group or organisation that is related to a 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.

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 Requirement System quality requirements constrain a system and its (development and usage) environment in order to support an activity, thus in order to satisfy a business goals. System quality requirements directly constrain by the use of measurements specific behavioural and structural 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. Synonyms: (system -specific) non-functional requirement, quality requirement.

System Vision The system vision defines the (sometimes contractual) basis for the following requirements specification. It 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 is a characteristic of a system and its development in which the requirements are clearly and unambiguously linked backward to their sources and rationales and forward to the artefacts that are produced during following constructive disciplines based on the requirements. 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.

Traceability Matrix The traceability matrix describes the requirements traces that are produced over the whole development life cycle, defining the backward traces and the forward traces of the requirements artefacts.

Underspecified Artefact A content item is underspecified if it is in a state being not conformant to the corresponding concept type, i.e. incomplete and / or inconsistent. t. An artefact type remains underspecified if the corresponding concept types do not underlie further refinements remaining at the level of (1) the business process hierarchy at the example of the business specification and / or (2) the information system service hierarchy at the example 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 an actor in interaction with the system to satisfy a business goal. 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. It consists of a unique ID and a brief description of its purpose. A use case is related to (1) an actor that participates within a structured scenario, (2) a structured scenario as an ordered sequnce of actions and interactions that occurs under certain (3) conditions that describe the state of the system before the scenario is executed (precondition) and the state after the execution (postcondition), (4) an action being a unit that describes a specific behaviour within a scenario and finally (5) an information system object that is the system-side representation of an information object being created or modified as a consequence of a system action.

Use Case Overview The use case overview 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. It defines the relationships between actors and use cases as well as between the use cases.

User Group A user group is a group of individuals with a specific proficiency directly performing a set of business tasks and interacting with information systems.

Bibliography

[Ale03] Ian Alexander. Misuse cases: Use cases with hostile intent. *IEEE Software*, pages 58–66, 2003.

[AW03] A. Aurum and C. Wohlin. The fundamental nature of requirements engineering activities as a decision-making process. *Information and Software Technology*, 45(14):945–954, 2003.

[AW05] A. Aurum and C. Wohlin. Aligning requirements with business objectives: a framework for requirements engineering decisions. In *Proceedings Requirements Engineering Decision Support Workshop*, 2005.

[BAB09] *A guide to the business analysis body of knowledge (BABOK Guide)*. International Institute of Business Analysis (IIBA), March 2009.

[BBK+78] Barry W. Boehm, John R. Brown, Hans Kaspar, Myron Lipow, Gordon J. Macleod, and Michael J. Merrit. *Characteristics of Software Quality*. North-Holland, 1978.

[BCK05] Len Bass, Paul Clements, and Rick Kazman. *Software architecture in practice*. Addison-Wesley, Boston and Munich, 2005.

[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.

[BEJ07] R. Buschermöhle, H. Eekhoff, and B. Josko. Success rate and factors of it - projects 2006 in germany. In Hamid R. Arabnia and Hassan Reza, editors, *Software Engineering Research and Practice*, pages 644–650. CSREA Press, 2007.

[Ber08] A. Berre. Service oriented architecture Modeling Language (SoaML) - Specification for the UML Profile and Metamodel for Services (UPMS). Object Management Group (OMG), August 2008.

[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 Software-intensiver Systeme. Technical Report TUM-I0816, Technische Universität München, June 2008.

[BGN06] Michael Brenner, Markus Garschhammer, and Friederike Nickl. Requirements engineering und it service management - ansatzpunkte einer integrierten sichtweise. In Heinrich C. Mayr and Ruth Breu, editors, *Modellierung*, volume 82 of *LNI*, pages 51–66. GI, 2006.

[BKM07] Manfred Broy, Ingolf Krüger, and Michael Meisinger. A formal model of services. *ACM Transactions on Software Engineering Methodology (TOSEM)*, 16(1), 2007.

[Boe79] Barry W. Boehm. Software engineering: As it is. In *ICSE 79: Proceedings of the 4th international conference on software engineering*, pages 11–21, Piscataway, NJ, USA, 1979. IEEE Press.

[Boe03] B. Boehm. Value-based software engineering. *ACM SIGSOFT Software Engineering Notes*, 28(2):1, 2003.

[Boe06] Barry W. Boehm. A view of 20th and 21st century software engineering. In *ICSE 06: Proceedings of the 28th international conference on Software engineering*, 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*. McGraw-Hill Osborne Media, 2009.

[Bri96] A. Brinkkemper. Method engineering: Engineering of information systems development methods and tools. *Information and Software Technology*, pages 275–280, 1996.

[Bro03] M. Broy. Service-oriented systems engineering: Modeling services and layered architectures. In *FORTE*, pages 48–61, 2003.

[Bro05] M. Broy. Service-oriented systems engineering: Specification and design of services and layered architectures. In *Engineering Theories of Software Intensive Systems*, pages 47–81. Springer-Verlag, 2005.

[Bro06] M. Broy. Requirements Engineering as a Key to Holistic Software Quality. *Lecture Notes in Computer Science*, 4263:24, 2006.

[BW07] B. Berenbach and T. Wolf. A unified requirements model; integrating features, use cases, requirements, requirements analysis and hazard analysis. In *Second IEEE International Conference on Global Software Engineering, 2007. ICGSE 2007*, pages 197–203, 2007.

[BWHW05] C. Braun, F. Wortmann, M. Hafner, and R. Winter. Method construction - a core approach to organizational engineering. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1295–1299. ACM, 2005.

[CCR+95] G. Campbell, M. Cochran, S. Rose, L.P. Gates, and K.A. Johnson. A tailorable process for systems engineering (version 01.00.05), 1995.

[cha] The standish group report — chaos. http://www.standishgroup.com/.

[CHS08] Andreas Classen, Patrick Heymans, and Pierre-Yves Schobbens. What is in a feature : A requirements engineering perspective. Technical report, 2008.

[Cle05] David Cleland. *Project Management Field Guide 2nd Edition with Microsoft*. John Wiley and Sons, Inc., 2005.

[CM78] Joseph P. Cavano and James A. McCall. A framework for the measurement of software quality. In *Proceedings of the software quality assurance workshop on Functional and performance issues*, pages 133–139, 1978.

[CNYM99] Lawrence Chung, Brian A. Nixon, Eric Yu, and John Mylopoulos. *Non-Functional Requirements in Software Engineering (THE KLUWER INTERNATIONAL SERIES IN SOFTWARE ENGINEERING Volume 5) (International Series in Software Engineering)*. Springer, October 1999.

[Coc00] A. Cockburn. *Writing Effective Use Cases*. Addison-Wesley Professional, January 2000.

[Com99] President's Information Technology Advisory Commitee. Report to the president: Information technology research: Investing in out future, 1999.

[Cor04] MITRE Corporation. Guide to the (evolution) enterprise architecture body of knowledge.

http://www.mitre.org/work/tech_papers/tech_papers_04/04_0104/04_0104.pdf, Februrary 2004.

[Cru09] D. Cruz. *POSAAM: Eine Methode zu mehr Systematik und Expertenunabhängigkeit in der qualitativen Architekturbewertung*. PhD thesis, Technische Universität München, 2009.

[CW97] E. Conklin and W. Weil. Wicked Problems: Naming the Pain in Organisations, June 1997.

[Dam05] W. Damm. Controlling speculative design processes using rich component models. In *Fifth Conference on Application of Concurrency to System Design*, pages 118–119, 2005.

[Dav93] Alan M. Davis. *Software Requirements: Objects, Functions and States*. Prentice Hall PTR, New Jersey, 2nd edition, 1993.

[DeM79] T. DeMarco. *Structured Analysis and System Specification.* Prentice Hall International, 1979.

[DG96] G. Dinkhoff and V. Gruhn. *Geschäftsprozessmodellierung und Workflow-Management - Modelle, MEthoden, Werkzeuge*. In G. Vossen and J. Becker, 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)*. IEEE CS Press, 2009.

[DKK+05] Joerg Doerr, Daniel Kerkow, Tom Koenig, Thomas Olsson, and Takeshi Suzuki. Non-functional requirements in industry – three case studies adopting an experience-based NFR method. In *Proc. 13th International Conference on Requirements Engineering (RE'05)*, pages 373–382. IEEE CS, 2005.

[DKOA06] Joerg Doerr, Tom Koenig, Thomas Olsson, and Sebastian 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. 2003.

[(DT03] Business Modeling & Integration (BMI) Domain Task Force (DTF). Business process modeling notation. www.bpmi.org, August 2003.

[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. *Proceedings of Foundation of Interface Technologies*, 2005.

[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*. Dpunkt Verlag, 2008.

[ER03] A. Endres and H.D. Rombach. *A handbook of software and systems engineering: empirical observations, laws and theories*. Addison-Wesley, 2003.

[FGP04] A. Fleischmann, E. Geisberger, and M. Pister. Herausforderungen fuer das Requirements Engineering eingebetteter Systeme. Technical Report TUM-I4014, Technische Universität München, 2004.

[FHKS08] Jan Friedrich, Ulrike Hammerschall, Marco Kuhrmann, and Marc Sihling. *Das V-Modell XT (Informatik im Fokus)*. Springer, 2008.

[Fle08] Andreas Fleischmann. *Modellbasierte Formalisierung von Anforderungen für eingebettete Systeme im Automotive-Bereich*. PhD thesis, Technische Universitätt München, 2008.

[FSC06] A. Feller, D. Shunk, and T. Callarman. Value chains versus supply chains. www.ceibs.edu/knowledge/papers/images/20060317/2847.pdf, 2006.

[GBB+06] Eva Geisberger, Manfred Broy, Brian Berenbach, Juergen Kazmeier, Daniel Paulish, and Arnold Rudorfer. Requirements engineering reference model (rem). Technischer Bericht, Technische Universität Müunchen, 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 *Requirements Engineering, Proceesings of the First International Conference on Requirements Engineering*, pages 94–101, 1994.

[Gli05] Martin Glinz. Rethinking the notion of non-functional requirements. In *Proc. Third World Congress for Software Quality*, volume II, pages 55–64, 2005.

[Gli07] Martin Glinz. On Non-Functional Requirements. In *Proceedings of the 15th IEEE International Requirements Engineering Conference*, 2007.

[Gro] The Capgemini Group. Integrated architecture framework. www.capgemini.com/iaf.

[Gro00] The Business Rules Group. Defining business rules — what are they really? www.businessrulesgroup.org, July 2000.

[Gro06] The Open Group. The open group architecture framework (togaf). http://opengroup.org/architecture, 2006.

[GW06] T. Gorschek and C. Wohlin. Requirements abstraction model. *Requir. Eng.*, 11(1):79–101, 2006.

[Hos87] W. A. Hosier. Pitfalls and safeguards in real-time digital systems with emphasis on programming. In *ICSE 87: Proceedings of the 9th international conference on software engineering*, pages 311–327, Los Alamitos, CA, USA, 1987. IEEE Computer Society Press.

[HP08] Andrea Herrmann and Barbara Paech. Moqare: misuse-oriented quality requirements engineering. Secaucus, NJ, USA, 2008.

[HT09] B. Hummel and J. Thyssen. Behavioural specification of reactive systems using steam-based i/o tables. In *7th IEEE International Conference on Software Engineering and Formal Methods*, 2009.

[HWFP07] Colin Hood, Simon Wiedemann, Stefan Fichtinger, and Urte Pautz. *Requirements Management: Interface Between Requirements Development and all other Engineering Processes*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[IJH09] S. Islam, M. Joarder, and S. Houmb. Goal and risk factors for offshore outsourced software development from vendor's viewpoints. *International Conference on Global Software Engineering*, 0:347–352, 2009.

[Isl09] S. Islam. Software development risk management model - a goal driven approach. In *7th joint meeting of the European Software Engineering Conference and the ACN SIGSOFT Symposium on the Foundation of Software Engineering*. ACM Press, 2009.

[ISO03] ISO 9126: Product Quality – Part 1: Quality Model, 2003.

[(IT99] International Telecommunication Union (ITU). Message sequence chart, 1999.

[Jac01] Michael Jackson. *Problem frames: analyzing and structuring software development problems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[JBR99] Ivar Jacobson, Grady Booch, and James Rumbaugh. *The unified software development process*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[JE03] L. Jiang and A. Eberlein. Decision support for requirements engineering process development. In *Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on*, volume 2, 2003.

[Kat09] Chrstina Katz. Development of an artifact model for measuring the degree of completion in it-projects. Master's thesis, Technische Universität München, 2009.

[KCH+90] K. Kang, S. Cohen, J. Hess, W. Nowak, and S. Peterson. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. 1990.

[KH08] M. Kuhrmann and U. Hammerschall. Anpassung des V-Modell XT - Leitfaden zur organisationsspezifischen Anpassung des V-Modell XT. Projektbericht, Technische Universit/"at M/"unchen, 2008.

[KK03] P. Kroll and P. Kruchten. *The rational unified process made easy: a practitioner's guide to the RUP*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[KKZB08] N. Koch, A. Knapp, G. Zhang, and H. Baumeister. UML-based Web Engineering: An Approach based on Standards. *Koch, N. and Knapp, A. and Zhang, G. and Baumeister, H.*, 2008.

[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 Softw.*, 13(1):12–21, 1996.

[KR97] Joachim Karlsson and Kevin Ryan. A cost-value approach for prioritizing requirements. *IEEE Softw.*, 14(5):67–74, 1997.

[KS96] G. Kotonya and I. Sommerville. Requirements engineering with viewpoints. *Software Engineering Journal*, 11, 1996.

[MFsA09] D. Mendez Fernandez and Capgemini sd&m AG. Requirements engineering method v 1.1. Available on Demand at Capgemini sd&m AG, 2009.

[oEEE98] The Institute of Electrical and Inc. Electronics Engineers. *IEEE Recommended Practice for Software Requirements Specifications*. IEEE Std 830-1998, 1998.

[oEEE00] The Institute of Electrical and Inc. Electronics Engineers. IEEE-Std-1471-2000 Recommended Practice for Architectural Description of Software-Intensive Systems. IEEE-SA Standards Board, September 2000.

[oGCO07] Office of Government Commerce (OGC). *The Introduction to the ITIL Service Lifecycle Book*. TSO (The Stationery Office), Norvich, UK, 2007.

[OMG07a] OMG. Unified modeling language (uml): Infrastructure – version 2.1.1. Technical report, Object Management Group, 2007.

[OMG07b] OMG. Unified modeling language (uml): Superstructure – version 2.1.1. Technical report, Object Management Group, 2007.

[Pal91] J. D. Palmer. Issues of traceability in integrating tools. *Proc. IEE Colloquium Tools and Techniques for Maintaining Traceability during Design*, 1991.

[PEM03] F. Paetsch, A. Eberlein, and F. Maurer. Requirements engineering and agile software development. In *Proceedings of the Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, page 308. IEEE Computer Society Washington, DC, USA, 2003.

[Poh07] Klaus Pohl. *Requirements Engineering. Grundlagen, Prinzipien, Techniken*. Dpunkt Verlag, 2007.

[RJ01] B. Ramesh and M. Jarke. Toward reference models for requirements traceability. *IEEE Trans. Softw. Eng.*, 27(1):58–93, 2001.

[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 *First Swedish Conference on Software Engineering Research and Practise: Proceedings*, 2001.

[RR99] S. Robertson and J. Robertson. *Mastering the Requirements Process*. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 1999.

[RR07] J. Robertson and S. Robertson. Volere Requirements Specification Templates - Edition 11. Available Online at 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] Chris Rupp. *Requirements Engineering und Management: Professionelle, Iterative Anforderungsanalyse für die Praxis*. Carl Hanser Verlag, 2007.

[Sch91] A.-W. Scheer. Architektur integrierter informationssysteme—grundlagen der modellierung. *Berlin et al*, 4, 1991.

[Sch01] B. Schätz. The odl operation definition language and the autofocus/quest application framewoek aqua. Technical report, Technische Universit/"at M/"unchen, 2001.

[Sch02] A.-W. Scheer. *ARIS. Vom Geschäftsprozess zum Anwendungssystem*, volume 4. Springer-Verlag, 2002.

[Sch04] J. Schekkerman. *How to Survive in the Jungle of Enterprise Architecture Frameworks: Creating or Choosing an Enterprise Architecture Framework*. Trafford Publishing, 2004.

[Sch08] B. Schätz. *Model-Based Development of Software Systems: From Models to Tools*. Habilitation, Technische Universität München, 2008.

[SEH09] F. Salger, G. Engels, and A. Hofmann. Inspection effectiveness for different quality attributes of software requirement specifications: An industrial case study. In *Proceedings of the 7th international workshop on Software Quality (WoSQ 09)*. IEEE CS Press, 2009.

[SFGP05] B. Schätz, A. Fleischmann, E. Geisberger, and M. Pister. Modebased requirements engineering with autoraid. In Armin B. Cremers, Rainer Manthey, Peter Martini, and Volker Steinhage, editors, *Workshop "Modellbasierte Qualitätssicherung" (QUAM)*, Lecture Notes in Informatics (LNI), pages 511–516. Bonner Köllen Verlag, 2005.

[SG02] Richard Stallman and Joshua Gay. *Free Software, Free Society: Selected Essays of Richard M. Stallman*. Free Software Foundations, June 2002.

[SJ96] A.-W. Scheer and W. Jost. Gesch ”aftsprozessmodellierung innerhalb einer Unternehmensarchitektur. *Gesch ”aftsprozessmodellierung und Workflow-Management*, pages 29–49, 1996.

[Smi] Larry Smith. Project clarity through stakeholder analysis.

[SPHP02] B. Schätz, A. Pretschner, F. Huber, and J. Philipps. Model-Based Development of Embedded Systems. In *Advances in Object-Oriented Information Systems, Lecture Notes in Computer Science*, volume 2426, pages 298–311. Springer-Verlag, 2002.

[SS06] A. Sillitti and G. Succi. Requirements Engineering for Agile Methods. *Engineering and Managing Software Requirements*, pages 309–325, 2006.

[Ste90] G. Stehle. Requirements traceability for real-time systems. In *Proc. EuroCASE II*, April 1990.

[Thu04] V. Thurner. *Formal fundierte Modellierung von Geschäftsprozessen*. PhD thesis, Technische Universität München, April 2004.

[TK09] T. Ternité and M. Kuhrmann. Das V-Modell XT 1.3. Metamodell. Forschungsbericht TUM-I0905, Technische Universität München, March 2009.

[TOG09] Ontologies for SOA The Open Group. Soa ontology, 2009.

[vL03] Axel van Lamsweerde. From system goals to software architecture. In M. Bernardo and P. Inverardi, editors, *School on Formal Methods*, volume LNCS 2804, pages 25–43, Bertinoro, Italy, 2003. Springer-Verlag.

[vL04] A. van Lamsweerde. Goal-oriented requirements enginering: a roundtrip from research to practice. In *12th IEEE International Requirements Engineering Conference*, pages 4–7, 2004.

[vL09] Axel van Lamsweerde. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley, March 2009.

[WDW08] Stefan Wagner, Florian Deissenboeck, and Sebastian Winter. Managing quality requirements using activity-based quality models. In *WoSQ '08: Proceedings of the 6th international workshop on Software quality*, 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*. Wiley & Sons, 3 edition, 2002.

[Wie03] Karl Eugene Wiegers. *Software Requirements*. Microsoft Press, Redmond, WA, USA, 2003.

[WMFIL09] S. Wagner, D. Mendez Fernandez, S. Islam, and K. Lochmann. A security requirements approach for web systems. In *Proc. Workshop Quality Assessment in Web (QAW 2009)*, 2009.

[Wri91] S. Wright. Requirements traceabilitywhat? why? and how? In *IEE Colloquium on Tools and Techniques for Maintaining Traceability During Design*, 1991.

[Zac87] J.A Zachman. A framework for information systems architecture. IBM Systems Journal, 1987.

[ZYCJ06] T. Zhang, S. Ying, S. Cao, and X. Jia. A modeling framework for service-oriented architecture. In *Quality Software, 2006. QSIC 2006. Sixth International Conference on*, pages 219–226, 2006.