Chapter 4
The Requirements Engineering Framework

This chapter provides an overview of our requirements engineering framework. The framework defines:

- *Four facets of the system context: the subject, the usage, the IT system, and the development facets*
- *Three core requirements engineering activities: documentation, elicitation, and negotiation*
- *Two cross-sectional activities: validation and management*
- *Three kinds of requirements artefacts: goals, scenarios, and solution-oriented requirements*

The structure of this book is derived from the requirements engineering framework and is briefly described at the end of this chapter.

K. Pohl, *Requirements Engineering*,

4.1 Goal of Requirements Engineering: Establishing a Vision in Context

Vision defines intended change

Each requirement engineering process starts with an aim to change the current reality. Regardless of the complexity of the project, the essence of the desired change should be defined briefly and precisely. We call this definition of the envisioned change the system "vision". A prominent example is the vision expressed by John F. Kennedy in 1961 when he said: "First, I believe that this nation should commit itself to achieving the goal, before this decade is out, of landing a man on the moon and returning him safely to the earth" (speech of J.F. Kennedy in 1961 [Dudley 2000]).

A vision states a goal and not how the goal should be achieved

A vision may also express a small change to the current reality, such as the integration of a new functionality into a multimedia system, or the increase of the level of security of an online banking system. A vision defines only what should be changed without stating how the change should be implemented. In other words, the vision defines a goal but does not state how this goal should be achieved. In the example of John F. Kennedy, the vision defines that a man has to be sent to the moon and that he has to be brought back safely without saying anything, for example, about the transportation to be used, how to land, or how to get back to Earth again.

Vision as guidance

A vision does not express an unachievable illusion. Rather, it describes a goal that is clearly defined and verifiable, and often also associated with a particular point in time when it should be achieved. The vision serves, for all stakeholders involved in the development process, as guidance throughout the entire development process. The stakeholders align their activities with the defined vision. The information expressed in the vision is not sufficient to define all the requirements for the system at the required level of detail. The additional information needed to define the requirements fully must thus be elicited from other requirement sources such as stakeholders (e.g. customers, system users, domain experts), existing documents (e.g. laws, guidelines, standards), and existing systems (e.g. legacy systems, competitors' systems); see Part IV.b.

The system context

Each software-intensive system is embedded within a given context (the "system context"; see Glossary) that contains the requirement sources and strongly influences the definition of the system requirements. We elaborate on the system context and its influence on the system requirements in Part II.

Establishing a vision in context

The vision and the system context are thus the two essential inputs for the requirements engineering process. Whereas the vision is typically clearly defined, the system context is often not fully known and understood at the beginning of the requirements engineering process. The main goal of requirements engineering is thus to "establish a vision within an existing context" (see, e.g. [Jarke and Pohl 1993; Pohl 1997]).

4.2 Overview of the Framework

Our requirements engineering framework (see Fig. 4-1) defines the major structural elements of a requirements engineering process required for establishing a vision within an existing context. The framework consolidates various research results which have been developed based on problem statements elicited from industrial practice and which have been successfully validated and transferred to industry.

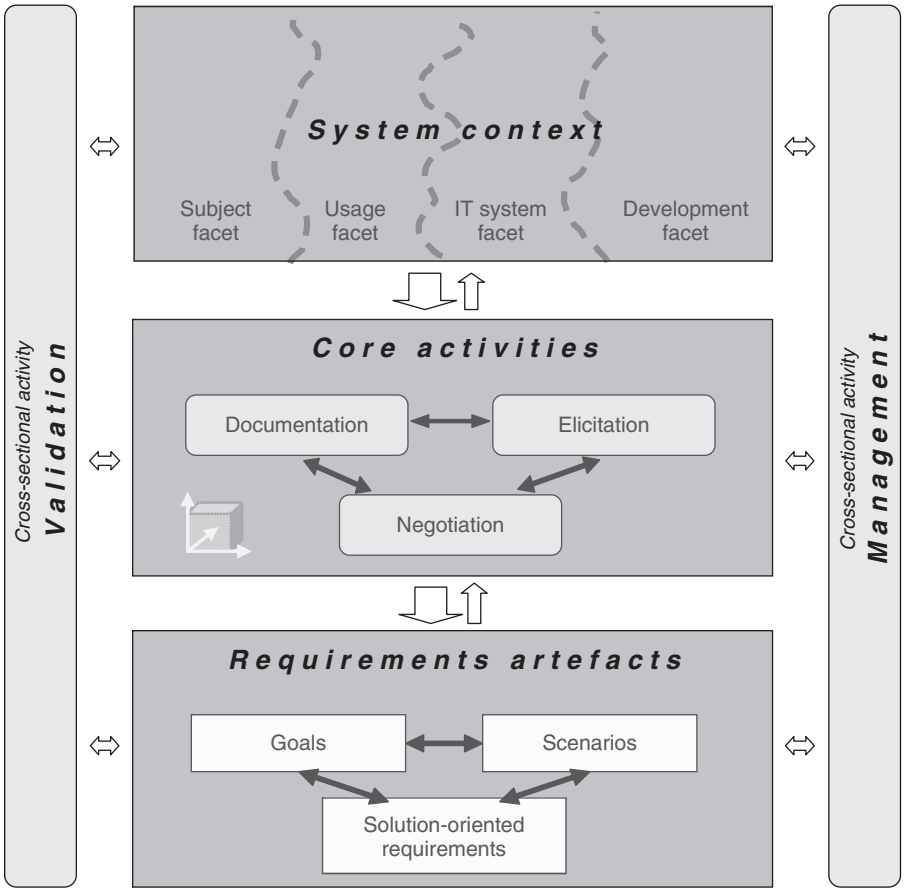

Fig. 4-1 *The requirements engineering framework*

Our framework has been adopted in several organisations

Our framework has been successfully introduced in a number of organisations and companies which use the framework as a reference for structuring their requirements engineering processes, as a reference for the training of managers, requirements engineers, and developers, and for analysing the strengths and weaknesses of their requirements engineering processes. The framework consists of the following main building blocks:

Four context facets

- *System context*: The framework structures the system context into four parts: the subject, the usage, the IT system, and the development facet.

Documentation, elicitation and negotiation

- *Three core requirements engineering activities*: The three core requirements engineering activities (elicitation, documentation, and negotiation) are performed iteratively in order to establish the vision within the existing context.

Validation and management

- *Two cross-sectional activities*: The two cross-sectional activities of validation and management support the core activities and secure the results of requirements engineering.

Three types of requirements artefacts

- *Requirements artefacts*: Our framework distinguishes three essential types of requirements artefacts: goals, scenarios, and solution-oriented requirements.

Details of the framework

In Sections 4.3–4.6, the elements of our framework and the relationships between these elements are described in more detail. In Parts II–VI, fundamentals, principles,

and techniques related to each element of the framework are presented. Section 4.7 provides an overview of the structure of the book, which is based on our requirements engineering framework.

4.3 Four Context Facets

Elicitation and consideration of all context aspects

The requirements for a software-intensive system are strongly influenced by the system context. A sufficient understanding of the context is an essential prerequisite for developing a good requirements specification. The system context comprises a large number of different aspects that are relevant to the system to be developed, such as business processes and workflows, existing hardware and software components, other systems that interact with the system, physical laws, safety standards, system users, customers — just to name a few. Due to the complexity demanded of the increasing system functionality and integration of systems, analysing the system context is a difficult task. Moreover, coupled with higher demands for system quality, reduced development costs and shorter time frames, consideration of the relevant aspects of the context becomes exceedingly difficult.

Insufficient consideration of context aspects leads to requirements defects

However, if relevant context aspects (see Definition 5-2, Page 65) are neglected, insufficiently considered, or documented inadequately, this may result in an incorrect and incomplete requirements specification. In order to support the systematic consideration and analysis of the relevant context aspects, our framework structures the system context into four facets[7] which have to be considered for each software-intensive system during requirements engineering, namely:

Objects and events relevant for the system

- *Subject facet*: The subject facet includes the objects and events in the system context that are relevant for the system. This includes, for example, objects and events that the system must store or process information about. In other words, information about the objects and events in the subject facet must be represented in the system. The objects of interest can be tangible as well as intangible objects. For instance, a software component that measures the speed of a vehicle requires a representation of the intangible object "speed" within the component. The subject facet also includes aspects that influence the representation of information about the objects and the events in the system, such as laws that forbid or regulate the recording of certain types of data within a software-intensive system, or laws which restrict the accuracy of the recorded data or the frequency of updating the data.

System usage

- *Usage facet*: A software-intensive system is used by people or other software-intensive systems in order to achieve a goal or to accomplish a certain task. The usage facet comprises all aspects concerning the system usage by people and other systems. This includes, for example, the various usage goals which exist, desired workflows, different user groups with specific characteristics, different interaction models with different associated interfaces as well as laws and standards restricting or influencing the system usage.

[7] The four facets are based on the four worlds of requirements engineering proposed in [Mylopoulos et al. 1990; Jarke and Pohl 1993].

- *IT system facet*: The system to be developed is eventually deployed into an existing IT infrastructure, which typically comprises existing software-intensive systems as well as existing hardware and software platforms, communication network(s), peripheral devices, and other hardware and software components used. The IT system facet comprises all aspects of the operational and technical environment including policies and strategies defining restrictions or guidelines for the use of any type of technology or operational environment. All these aspects of the technical and operational environment, as well as the constraints resulting from them, influence the definition of the system requirements. For example, the IT strategy might prescribe the communication protocol to be used, which obviously influences the communication requirements defined for the system, or the software platform might predefine a set of supported operating systems which in turn influences the defined requirements.

IT system environment

- *Development facet*: The development facet comprises all aspects of the context concerning the development process of the system. This includes process guidelines and constraints, development tools, quality assurance methods, maturity models, quality certifications, and other means or techniques for ensuring the quality (e.g. safety or security) of a software-intensive system. Each aspect of the development facet may be restricted by the client or by certain laws and standards. For instance, the client may demand that only certified tools are used during system development, or a standard might require that the system fulfils certain test criteria.

Aspects of the development process

4.3.1 Relationships between the Four Context Facets

The relationships between the four context facets can be illustrated by means of a very simple, logical model of a software-intensive system.

Relationship between subject facet and IT system facet

Each software-intensive system requires a representation of information about real-world objects (residing in the subject facet) such as the speed of a car or the name of a customer. The system represents the information about these real-world objects in a digital format using the available technology (residing in the IT system facet). The representation of the information about the real-world objects in the system constitutes a relationship between the subject facet and the IT system facet (see the black arrow labelled "Representation" in Fig. 4-2).

Relationship between IT system facet and usage facet

The system processes the represented information according to the defined functionality. The processing of the information takes place within the IT system facet. However, the system presents the results of the processing to its users (a person or another system) using an appropriate user interface. The presentation of the results to the user by means of some interface device constitutes a relationship between the IT system facet and the usage facet (see the black arrow labelled "Presentation" in Fig. 4-2).

Relationship between usage facet and subject facet

The system user interprets the output of the system obtained and associates it with real-world objects of the subject facet. This constitutes a relationship between the usage facet and the subject facet (see the black arrow labelled "Association" in Fig. 4-2).

Role of the development facet

The software-intensive system itself is a product of a development process which takes place in the development facet. The task of the development process is to consider, besides the relevant aspects of the development facet, the relevant aspects of the other three context facets and their relationships. Hence the development facet is related to each of the other three facets (as indicated by the three grey arrows in Fig. 4-2).

The four context facets and their relationships are discussed in detail in Part II.

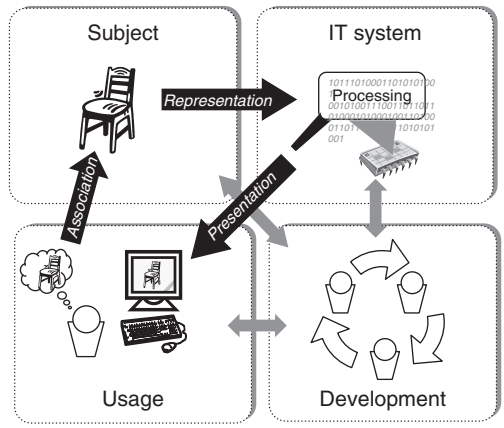

Fig. 4-2 *Logical relationships between the four context facets*

4.3.2 Use of the Four Context Facets

Improved quality of requirements documents

Structuring the system context into the four facets is a heuristic that has proven useful during, among other things, requirements elicitation, negotiation, and validation. Structuring the context into four facets supports a systematic consideration of the system context during requirements engineering. By considering each of the four facets, as well as their relationships, relevant context aspects can be identified more easily and, for example, the risk of neglecting an entire facet is reduced. Hence the completeness and the correctness of the specified requirements are significantly improved. According to our experience, the definition and use of simple checklists for each context facet, and for the relationships between the facets, facilitates a more systematic consideration of the system context and thus leads to requirements specifications of higher quality.

4.4 Three Core Activities

The requirements engineering process takes place in the development facet. The main goals of requirements engineering are characterised by the three dimensions of requirements engineering, as explained in Section 4.4.1 (see [Pohl 1993; Pohl 1994]).

From the three dimensions of requirements engineering, the three core activities of requirements engineering (documentation, elicitation, and negotiation) can be derived (see Section 4.4.2).

4.4.1 The Three Dimensions of Requirements Engineering

Figure 4-3 illustrates the three dimensions of requirements engineering, which can be characterised as follows:

- *Content dimension*: The content dimension deals with the understanding of the system requirements attained. At the beginning of the requirements engineering process, besides the system vision, only a few system requirements are known. The understanding of these requirements is typically vague, and the detailed requirements are mostly unknown. In contrast, at the end of the process, all the requirements are known and understood, preferably at the required level of detail. During the requirements engineering process, requirements must thus be elicited, including the development of new and innovative requirements. Consequently, the first essential goal of the requirements engineering process can be defined as: "All relevant requirements shall be explicitly known and understood at the required level of detail."

All requirements are known and understood in detail

- *Agreement dimension*: The agreement dimension deals with the level of agreement achieved between the relevant stakeholders about the known requirements. Different stakeholders can have different opinions about a system requirement. Conflicts between the stakeholders about system requirements thus have to be detected as early as possible. The detected conflicts should be resolved, either by achieving consensus or by making (well-founded) decisions. If requirements are defined without consolidating the stakeholders' different opinions, the unresolved conflicts will inevitably surface during or after the deployment of the new system. Unresolved conflicts put the acceptance of the system at risk and thereby endanger the realisation of the system vision. Consequently, the second goal of the requirements engineering process can be defined as: "To establish a sufficient agreement about the system requirements between the involved stakeholders."

Establish sufficient stakeholder agreement

- *Documentation dimension*: The documentation dimension deals with documenting and specifying the system requirements using different documentation/specification formats. Usually, information that is elicited during requirements engineering is first documented informally, either as a note, sketch, statement in a minute, or hand drawing, for example. Later, the requirements are documented and specified according to the documentation and specification formats and rules defined for the project. Such formats and rules can define the modelling language or a template to be used for documenting or specifying a particular type of requirement. The rules may also include other criteria for ensuring high quality of the requirements documentation and specification, such as consistency rules for requirements expressed in different formats. The third essential goal of the requirements engineering process can be defined as: "All requirements shall be documented/specified in compliance with the relevant documentation/specification formats and rules."

Documentation/ specification of requirements in compliance with the defined formats and rules

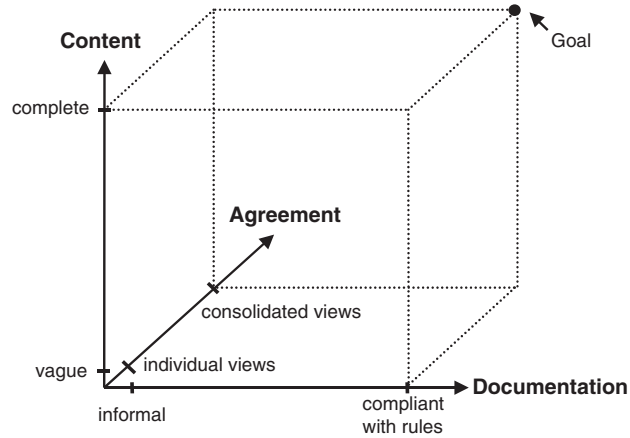

Fig. 4-3 *The three dimensions of requirements engineering (based on [Pohl 1994])*

The three essential goals of requirements engineering can thus be defined as follows:

Definition 4-1: *Goals of requirements engineering*

Requirements engineering is a cooperative, iterative, and incremental process which aims at ensuring that:

(1) All relevant requirements are explicitly known and understood at the required level of detail.
(2) A sufficient agreement about the system requirements is achieved between the stakeholders involved.
(3) All requirements are documented and specified in compliance with the relevant documentation/specification formats and rules.

4.4.2 The Core Activities

Derivation of the core activities from the three dimensions

From the three dimensions of requirements engineering, the three core activities of requirements engineering can be derived. Each core activity significantly contributes to the achievement of one of the three sub-goals of requirements engineering. We explain the three core activities below. The interactions between the core activities, as well as between the core activities and the cross-sectional activities, are explained in Section 4.5.1.

Documentation

Compliance with documentation and specification rules

The focus of the documentation activity is the documentation and specification of the elicited requirements according to the defined documentation and specification rules. In addition, other important types of information such as rationale or decisions must

be documented (see Section 15.1). The documentation activity thus distinguishes the following sets of rules:

- *General documentation rules*: These rules apply to all kinds of information to be documented such as interview and meeting protocols, information about the context or decisions and rationale. These rules define, for instance, the document layout, required document headers, and required document management information such as authors or a version history.
- *Documentation rules*: These rules apply to each requirement documented at different stages of the requirements engineering process. The rules aim to ensure a sufficient quality of the documentation of the requirements mainly for use in other requirements engineering activities (e.g. negotiation or validation) while at the same time keeping the documentation effort low. Documentation rules may, for instance, prescribe a specific template to be used for documenting the requirements.
- *Specification rules*: These rules apply to all requirements which are included in the requirements specification. The specification rules aim to ensure high quality of the specified requirements which are used in the subsequent development activities as key input or might be part of contracts. The specification rules may prescribe, for instance, the use of syntactic requirements patterns or a requirements specification language (see Definition 17-6).

In general, specification rules are typically more restrictive than documentation rules.

Use of different formats

Depending on the intended use of a requirements artefact, different documentation and specification formats can be used. For example, a requirement might be documented using natural language to facilitate communication with a typical end user, while at the same time be specified using a formal requirements language to support the system architect in defining the system architecture. In general, different stakeholders prefer different documentation/specification formats. Hence a requirements artefact may have to be translated from one format into another one. The issue becomes keeping the documentation/specification of a requirement held in different formats consistent across formats when undertaking any change. The documentation activity is discussed in more detail in Part IV.a.

Elicitation

Progress in the content dimension

The goal of the elicitation activity is to improve the understanding of the requirements, i.e. to achieve progress in the content dimension. During the elicitation activity, requirements are elicited from stakeholders and other requirement sources. In addition, new and innovative requirements are collaboratively developed.

Eliciting requirements and jointly developing innovative requirements

The requirement sources relevant for the system are not always known at the beginning of the process. An essential task of the elicitation activity is therefore the systematic identification of relevant requirement sources. Relevant requirement sources include the stakeholders involved in the process, existing documentation, and existing predecessor systems. Requirements are elicited, for example, by interviewing the stakeholders or by analysing existing documents and systems. In addition, innovative requirements (which can typically not simply be elicited from a requirement source) are developed in a collaborative and creative process. The development of innovative requirements can, for example, be supported by applying creativity

techniques such as brainstorming. The elicitation activity is described in more detail in Part IV.b.

Negotiation

Individual, conflicting stakeholder opinions

The system has to fulfil the needs and wishes of different stakeholders. Obviously, the needs and wishes of the different stakeholders can vary. Each stakeholder has his/her own view about the system to be developed. The different opinions of the stakeholders can be in conflict with one another.

Resolution of all existing conflicts

The goal of the negotiation activity is therefore twofold: First, all conflicts between the viewpoints of the different stakeholders have to be detected and made explicit. Second, the identified conflicts should be resolved (as far as possible). Depending on the cause of the conflict, different strategies can be applied for resolving it. At the beginning of the requirements engineering process, typically the viewpoints of the different stakeholders differ significantly. Ideally, at the end of the requirements engineering process, the negotiation activity has identified and resolved all conflicts which exist between the different stakeholders involved. The negotiation activity is described in more detail in Part IV.c.

4.5 Two Cross-Sectional Activities

Besides the three requirements engineering core activities, two cross-sectional activities significantly influence the requirements engineering process. The cross-sectional activities support the three core activities and secure the results of requirements engineering. The two cross-sectional activities are described in the following sub-sections.

Validation

The validation activity is a cross-sectional activity which consists of three sub-activities:

Validation of the artefacts

- *Validation of the requirements artefacts:* The validation of the requirements artefacts aims at detecting defects in the requirements. Only requirements with high quality provide a sound basis for the architectural design, the implementation of the system, and the development of test artefacts. Defects in requirements entail defects in the architecture, in the implementation, and in test artefacts. Requirements defects are rarely caused only by insufficient or inappropriate requirements documentation. In many cases, defects can be attributed to the non-fulfilment of one or more goals in each of the three dimensions (see Section 4.4.2). Validation thus has the key aim of validating the artefacts with regard to the content, the documentation, and the agreement dimensions. A requirement should only be used as a reference for the further development process or as part of a (legal) contract if it has been successfully validated under the consideration of all three validation aspects (content, documentation, and agreement).

Validation of the activities

- *Validation of the core activities:* The validation of the core activities (documentation, elicitation, and negotiation) has the goal of checking the compliance between

the activities performed and the process and/or activity specifications. For example, one should validate whether the steps defined for an activity and the defined follow-up activities have been performed.

- *Validation of the consideration of the system context:* The validation of the consideration of the system context aims at validating whether the system context has been considered in the intended way during requirements engineering. In other words, this sub-activity aims at validating whether all relevant stakeholders have been involved in the process at the right time and whether the relevant context aspects of all four context facets have been considered during the requirements engineering process. For example, with respect to the usage facet, one should validate whether all required interactions between the system and its actors (users and other systems) have been elicited.

Validation of context consideration

The goals, principles, and techniques of the validation activity are described in detail in Part V.

Management

The management activity can be subdivided into the following three essential sub-activities:

- *Management of the requirements artefacts:* This activity comprises the management of the requirements artefacts throughout the system lifecycle. The management of the requirements artefacts includes the prioritisation of requirements, the persistent recording of requirements (e.g. by storing them in a database), the configuration management and the change management of the requirements as well as maintaining requirements traceability.

Management of the artefacts

- *Management of the activities:* This activity comprises the planning and control of the requirements engineering activities in order to ensure an efficient and effective overall requirements engineering process. If necessary, the planned workflow of the activities can be aligned to accommodate the current project situation.

Management of the activities

- *Observation of the system context:* The observation of the system context aims at identifying changes in the system context that are relevant for the system. A relevant context change typically requires the execution of one or more requirements engineering activities or a re-scheduling of activities. For example, it might require the execution of an elicitation activity and a documentation activity in order to document the new requirements caused by this change, or the execution of a change management activity to adjust existing requirements accordingly.

Management of changes to the context

The three essential requirements management activities are described in detail in Part VI.

4.5.1 Interrelations between the Five Activities

There are obvious interactions between the activities defined in our requirements engineering framework, i.e. the three core activities and the two cross-sectional activities. For example, the execution of one core activity (and thus progress mainly in

one of the three dimensions) may lead to a decrease in the progress established in one of the other dimensions, and thus require the execution of additional activities. In other words, performing one requirements engineering activity typically causes the execution of additional requirements engineering activities. We illustrate the interactions between the three core and two cross-sectional activities in Examples 4-1 to 4-3.

Example 4-1: Elicitation of additional requirements

During an interview (i.e. the execution of an elicitation activity), new requirements are identified and documented in the interview minutes. The identification of the new requirements obviously leads to progress in the content dimension. However, the documentation of the new requirements in the interview minutes is not in compliance with the project-specific documentation rules. Thus an additional task for the documentation dimension is created, namely the documentation of the new requirements so as to be in compliance with the defined rules. In addition, the new requirements should be agreed between the stakeholders involved. Thus, a new validation activity is performed to check whether the stakeholders agree with the new requirement. During the validation of the agreement, conflicts about the requirement between the involved stakeholders might be identified. If so, these conflicts need to be resolved and the outcome of the conflict resolution must be documented and comply with the documentation rules, etc.

Example 4-2: Detection of a missing requirement

While reviewing a set of requirement artefacts (i.e. during the execution of a validation activity), the stakeholders detect that an important requirement has been omitted. The stakeholders briefly sketch the omitted requirement. Obviously, the new requirement is not yet documented in compliance with the defined documentation rules. Moreover, the documentation of the new (previously omitted) requirement does not contain all the required information, and not all the stakeholders have yet agreed to the new requirement. The identification of a new requirement during the validation activity (progress in the content dimension) thus might lead to the execution of additional elicitation, documentation, and negotiation activities.

Example 4-3: Removal of a requirement from the specification

Negotiations between customers and system users result in the removal of a requirement from the specification. The elimination of this requirement requires an evaluation of whether other requirements artefacts are affected by this change. The related requirements artefacts thus have to be analysed. In this example, the resolution of a conflict (progress in the agreement dimension) leads to the execution of additional activities in the content and documentation dimensions in order to check for inconsistencies resulting from the removal of the requirement and to adjust the documented artefacts accordingly, if required.

4.6 The Three Kinds of Requirements Artefacts

Goals, scenarios, and solution-oriented requirements

We use the term "requirements artefact" to refer to a documented requirement (see Definition 2-2 on Page 16). A requirements artefact thus documents a requirement using a specific documentation format. Different documentation formats which might be used to document a requirement are discussed in detail in Part III. In our framework, we differentiate three kinds of requirement artefacts, namely goals, scenarios, and solution-oriented requirements, which are also described in detail in Part III.

4.6.1 Goals

Stakeholder intentions

Requirements engineers need to understand the stakeholders' intentions with regard to the system to be developed. In requirements engineering, the stakeholders' intentions are documented as goals. Antón states in [Antón 1996] that goals "are high-level objectives of the business, organisation, or system". According to [Van Lamsweerde 2001], a goal is "an objective the system under consideration should achieve". We define a goal (in requirements engineering) as follows:

Definition 4-2: *Goal*

A goal is an intention with regard to the objectives, properties, or use of the system.

Prescriptive statements

Goals have a prescriptive nature, i.e. a goal states what is expected or required from the system. Thereby, goals differ from descriptive statements such as statements about the domain of the system (e.g. the description of a physical law). Example 4-4 depicts two goals for a navigation system.

Example 4-4: Goals for the car navigation system example

G_1: The system shall guide the driver to a desired destination automatically.
G_2: The response times of the system shall be 20% lower compared with the predecessor system.

Intentions of stakeholders

Goals (see Chapter 7) document the intentions of the stakeholders and abstract from system usage as well as from the realisation of the system. Goals refine the system vision into objectives to be fulfilled by the system.

A goal should be solution free, i.e. it should not predefine a specific solution. Hence, the stakeholders typically have many different alternatives for satisfying a goal, where each alternative may lead to different requirements.

Benefit of goals

The explicit definition of goals (stakeholder intentions) supports conflict resolution, leads to a better understanding of the system, and increases the acceptance of the system.

In Part III, we elaborate on different types of goals, the documentation of goals, and their usage in requirements engineering.

4.6.2 Scenarios

Examples of system usage

A scenario typically documents a concrete example of system usage. It thus illustrates the fulfilment (or non-fulfilment) of a goal (or set of goals). Thus, a scenario describes a concrete example of either how the system satisfies a goal or how it fails to satisfy a goal. A scenario may define an interaction sequence at different levels of abstraction. For example, a scenario can describe the interactions in detail and thus very close to reality or only document the essential interactions (and thereby abstract from the incarnation). We define a scenario as follows:

Definition 4-3: *Scenario*

A scenario describes a concrete example of satisfying or failing to satisfy a goal (or set of goals). It thereby provides more detail about one or several goals. A scenario typically defines a sequence of interaction steps executed to satisfy the goal and relates these interaction steps to the system context.

Scenarios illustrate goal satisfaction

Example 4-5 presents a scenario that documents a sequence of interactions between a driver and a driver assistance system. The scenario describes how the goal "facilitate automatic braking manoeuvres" can be achieved. In principle, goals and scenarios are complementary. For example, goals stimulate the elicitation of scenarios and vice versa.

In Part III.b, we elaborate on the characteristics of scenarios, different types of scenarios, their documentation, their usage in requirements engineering as well as their interrelations with goals.

Example 4-5: Scenario "Automatic braking manoeuvre"

Carl drives his car on the motorway at a speed of 50 mph. Peter, the driver of the car ahead of Carl, steps on the brake pedal firmly. After recognising that the car in front is braking, Carl pushes on the brake pedal as well. The on-board computer of Carl's vehicle detects that the safety distance to Peter's car is no longer maintained and issues a warning to the driver. The distance between the two cars continuously decreases. In order to support the driver, the on-board computer initiates an automated full braking. The computer informs Carl about the automatic braking manoeuvre. After the distance between the two cars stops decreasing, the on-board computer terminates the full braking manoeuvre. The on-board computer continues controlling the speed of Carl's car until the safety distance to Peter's car is maintained and informs Carl about the end of this "manoeuvre".

4.6.3 Solution-Oriented Requirements

Data, functions, behaviour, quality, and constraints

Solution-oriented requirements define the data perspective, the functional perspective, and the behavioural perspective on a software-intensive system (see

Section 2.2.1). Furthermore, solution-oriented requirements comprise (solution-oriented) quality requirements (see Section 2.2.2) and (solution-oriented) constraints (see Section 2.2.3).

Conceptual solution

In contrast to goals and scenarios, which should be defined fairly independently from a specific and intended solution, the definition of solution-oriented requirements often implies a conceptual (or logical) solution for the system (see Chapter 13). Data models, for instance, define entities, attributes, and relationships between entities (see Section 14.1).[8] Data models determine which data shall be represented in the system and, to some extent, how these data shall be represented. A behavioural model defines the states of the system and the externally visible behaviour with respect to these states (see Section 14.3). Thereby, it partially defines the intended solution. Solution-oriented requirements models thus often define a (partial) solution or even the basis for generating a solution from the requirements models (see Section 13.1).

In Part III.c, we present requirements models for documenting the data perspective, the functional perspective, and the behavioural perspective.

4.6.4 Use of the Three Kinds of Requirements Artefacts

The three artefact types are complementary

The three different kinds of requirements artefacts (goals, scenarios, and solution-oriented requirements) are used complementarily during requirements engineering. Using all three types of artefacts offers several advantages, as outlined throughout this book. For example, developing goals and scenarios prior to or along with solution-oriented requirements is an established principle for developing detailed system requirements based on a system vision (see e.g. [Jarke and Pohl 1993; Van Lamsweerde 2001; Antón 1996; Antón and Potts 1998; Yu 1997]). Applying a goal- and scenario-based approach typically leads to a significant improvement of the quality of the requirements specification. It improves, for example, the completeness of the specification (see amongst others [Antón and Potts 1998] as well as Section 7.1).

Moreover, scenarios put requirements into context and thus provide a good basis for deriving and developing detailed, solution-oriented requirements. For instance, scenarios explicitly document which stakeholders use the system as well as, via the goals associated with a scenario, the stakeholders' intentions for using the system (see Part III.b for more details). Moreover, goals and scenarios support the refinement of the requirements across different layers of abstraction, which is exploited by our goal- and scenario-based requirements engineering method COSMOD-RE (see Part VII).

4.6.5 The Term "Requirements" as Used in This Book

When we use the term "requirements" in this book, we refer to the three types of requirements outlined above (goals, scenarios, and solution-oriented requirements), as described in detail in Part III. For example, when we talk about requirements

[8] More precisely, a data model defines entity *types* and relationship *types* between the entity types (see Section 14.1).

elicitation we refer to the elicitation of goals, scenarios, and solution-oriented requirements. Therein, the term "solution-oriented requirements" refers to data, functional, and behavioural requirements as well as (solution-oriented) quality requirements and constraints.

D

Definition 4-4: *Requirements (as used in this book)*

When we use the term "requirements", we refer to goals, scenarios, and solution-oriented requirements (i.e. data, functional, and behavioural requirements, and solution-oriented quality requirements and constraints).

By defining more fine-grained types of requirements, a requirements classification scheme can be established which fits the needs of a particular domain, company, or project (see e.g. [Pohl 1996a; Young 2004]).

4.7 Overview of the Book

The structure of the book is derived from our requirements engineering framework. Fig. 4-4 illustrates which parts of the book discuss and describe which parts of the framework in detail.

Part II
- The structuring of the system context into four context facets is outlined in detail in Part II.

Part III
- The three kinds of requirements artefacts, their documentation, and their use are explained in detail in Part III. Part III is divided into three sub-parts. Part III.a focuses on goals, Part III.b on scenarios, and Part III.c on solution-oriented requirements.

Part IV
- The three requirements engineering core activities are explained in detail in Part IV. Part IV is divided into three sub-parts. Part IV.a outlines the documentation activity, Part IV.b the elicitation activity, and Part IV.c the negotiation activity.

Parts V and VI
- The two cross-sectional activities are explained in Part V (validation) and Part VI (management).

Parts VII and VIII

In addition to the detailed description of all aspects of our requirements engineering framework, we present our goal- and scenario-based requirements engineering method COSMOD-RE which integrates most of the aspects presented in this book (Part VII).

In Part VIII of this book, we outline the main challenges faced when applying our requirements engineering framework in a software product line setting and elaborate on the benefits of deriving test artefacts from requirements.

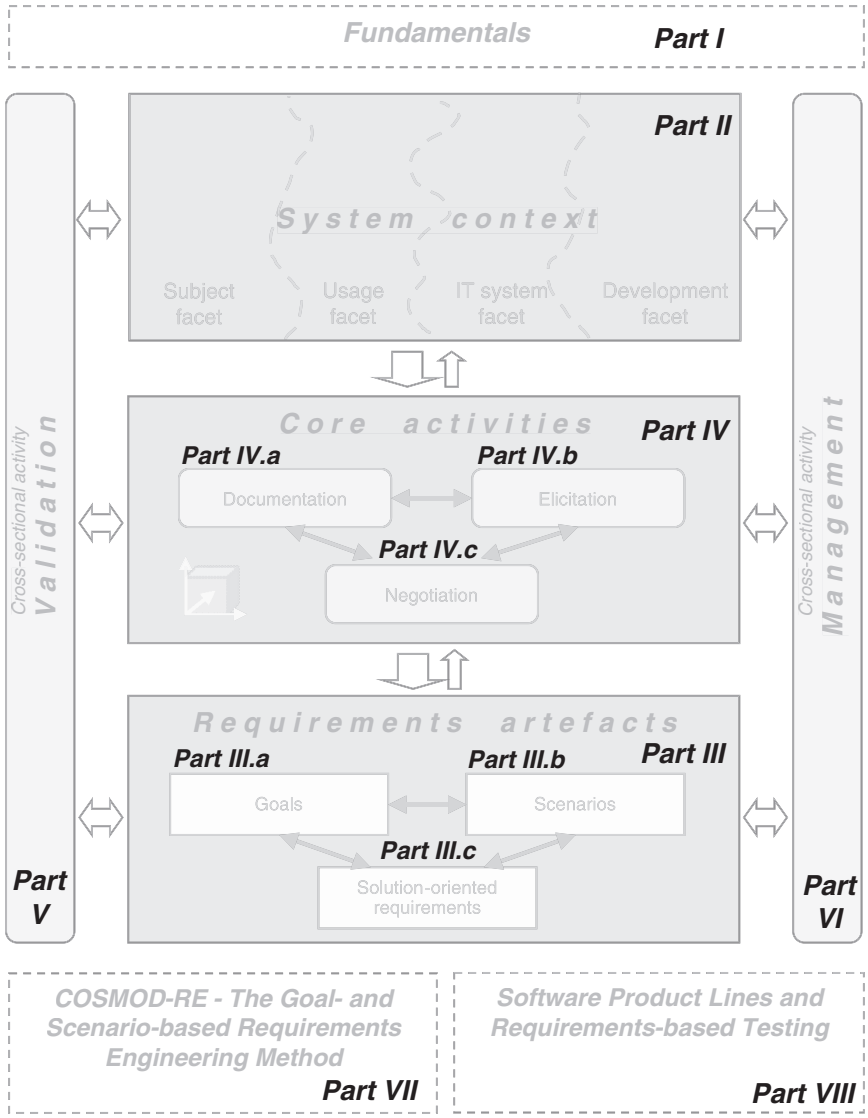

Fig. 4-4 *Structure of this book illustrated using the framework*

http://www.springer.com/978-3-642-12577-5

Requirements Engineering
Fundamentals, Principles, and Techniques
Pohl, K.
2010, XVII, 813 p., Hardcover
ISBN: 978-3-642-12577-5