7 Requirements Validation and Negotiation

Validation and negotiation during requirements engineering is meant to ensure that the documented requirements meet the predetermined quality criteria, such as correctness and agreement (see section 4.6). The introduced principles and techniques can be used to validate and negotiate individual requirements or entire requirements documents.

7.1 Fundamentals of Requirements Validation

During the requirements engineering activity, it is necessary to review the quality of the requirements developed. Among others, the requirements are presented to the stakeholders with the goal to identify deviations between the requirements defined and the stakeholders' actual wishes and needs.

Approving requirements

During requirements validation, the decision of whether a requirement possesses the necessary level of quality is made (see chapter 4) and whether the requirement can be approved to be used for further development activities (such as design, implementation, and testing). This decision should be made on the basis of predefined acceptance criteria.

Goal of validation

The goal of requirements validation is therefore to discover errors in the documented requirements. Typical examples of errors in requirements are ambiguity, incompleteness, and contradictions (see section 7.3).

Error proliferation

Requirements documents are reference documents for all further development activities. Therefore, errors negatively affect all further development activities. A requirements error that is discovered when the system is already deployed and operating requires all artifacts affected by the error to be revised, such as source code, test artifacts, and architectural descriptions. Correcting errors in requirements once the system is in operation therefore entails significant costs.

Legal risks

A contract between client and contractor is often based on requirements documents. Critical errors in requirements can lead to the fact that contractual agreements cannot be met, e.g., scope of supply and services, expected quality, or completion deadlines.

7.2 Fundamentals of Requirements Negotiation

Contradictory requirements cause conflicts.

If there is no consent among the stakeholders regarding the requirements and thus the requirements cannot be implemented collectively in the system, a conflict arises between the contradictory requirements as well as between the stakeholders that demand contradictory requirements. For example, one stakeholder could demand the system to shut down in case of a failure, whereas another stakeholder could require the system to restart.

Risks and opportunities of conflicts

The acceptance of a system is threatened by unresolved conflicts because unresolved conflicts cause the requirements of at least one group of stakeholders to not be implemented. In the worst case, a conflict causes stakeholder support to cease, causing the development project to fail (cf. [Easterbrook 1994]). Other than posing risks, conflicts can also be an opportunity for requirements engineering because conflicts between stakeholders require a solution that can potentially help discover new ideas for development and can illustrate different options (cf. [Gause and Weinberg 1989]). Therefore, treating and resolving conflicts openly during requirements engineering can increase acceptance.

Goal of requirements negotiation

The goal of negotiation is to gain a common and agreed-upon understanding of the requirements of the system to be developed among all relevant stakeholders.

Reducing costs and risks in late phases

Requirements validation and negotiation is an activity that must be performed (to a varying degree of intensity) throughout the entirety of requirements engineering. The validation and negotiation of requirements therefore causes additional effort and therefore additional costs. However, the advantages gained by performing requirements validation and negotiation as described in the previous sections (reduction of overall cost, increase in acceptance, supporting creativity and innovations) is usually significantly higher than the costs that arise due to the increased effort.

7.3 Quality Aspects of Requirements

A major aim of using quality criteria (e.g., completeness, understandability, agreement) in requirements validation is to be able to check requirements systematically (see section 1.1.2). In order to assure an objective and consistent validation, it is necessary that each quality criterion is concretized and refined. In correspondence with the overall goals of the requirements engineering process, the validation is carried out with the following goals:

- *Content:* Have all relevant requirements been elicited and documented with the appropriate level of detail?
- *Documentation:* Are all requirements documented with respect to the predetermined guidelines for documentation and specification?
- *Agreement:* Do all stakeholders concur with the documented requirements and have all known conflicts been resolved?

Three quality aspects

Each of the three goals implies an individual approach that focuses on specific aspects of the quality of the requirements. Therefore, the following three quality aspects have been defined:

- Quality aspect "content"
- Quality aspect "documentation"
- Quality aspect "agreement"

A requirement should be approved for further development activities only if all three quality aspects have been checked. The quality aspects are described in detail in the following sections and made concrete through different fine-grained quality criteria (with no claim of completeness).

7.3.1 Quality Aspect "Content"

The quality aspect "content" refers to the validation of requirements with respect to errors in the content. Errors in requirements with regard to content negatively influence the subsequent development activities and cause these activities to be based upon erroneous information.

Test criteria of the quality aspect "content"

Errors in requirements with regard to content are present when specific quality criteria for requirements (see section 4.6) or for requirements documents (see section 4.5) are violated. The validation of requirements with regard to the quality aspect "content" is successful once requirements

validation has been applied to the following error types and no significant shortcomings have been detected:

- *Completeness (set of all requirements):* Have all relevant requirements for the system to be developed (for the next system release) been documented?
- *Completeness (individual requirements):* Does each requirement contain all necessary information?
- *Traceability:* Have all relevant traceability relations been defined (e.g., to relevant requirements sources)?
- *Correctness/adequacy:* Do the requirements accurately reflect the wishes and needs of the stakeholders?
- *Consistency*: Is it possible to implement all defined requirements for the system to be developed jointly? Are there no contradictions?
- *No premature design decisions:* Are there any forestalled design decisions present in the requirements not induced by constraints (e.g., constraints that specifiy a specific client-server architecture to be used)?
- *Verifiability:* Is it possible to define acceptance and test criteria based on the requirements? Have the criteria been defined?
- *Necessity:* Does every requirement contribute to the fulfillment of the goals defined?

7.3.2 Quality Aspect "Documentation"

The quality aspect "documentation" deals with checking requirements with respect to flaws in their documentation or violations of the documentation guidelines that are in effect, such as understandability of the documentation formats and the consideration of organizational or project-specific guidelines regarding the documentation of requirements but also the structure of the requirements documents.

Implications of the violation of documentation guidelines

Ignoring the documentation guidelines can, among other things, lead to the following risks:

- *Impairment of development activities:* It may be impossible to carry out development activities that are based upon a specific documentation format.
- *Misunderstandings:* Requirements may not be understandable or may be misunderstood by the people that need to comprehend them. As a result, the requirement may be unusable.

- *Incompleteness:* Relevant information is not documented in the requirements.
- *Overlooking requirements:* If requirements are not documented at the position that they are supposed to in the requirements document, these requirements may be overlooked in subsequent activities.

Test criteria of the quality aspect "documentation"

Requirements validation with regard to the quality aspect "documentation" is successful when requirements validation has been applied to the following error types and no significant shortcomings have been detected:

Four test criteria of the quality aspect "documentation"

- *Conformity to documentation format and to documentation structures:* Are the requirements documented in the predetermined documentation format? For instance, has a specific requirements template or a specific modeling language been used to document the requirements? Has the structure of the requirements document been maintained? For instance, have all requirements been documented at the position defined by the document structure?
- *Understandability:* Can all documented requirements be understood in the context given? For instance, have all terms used been defined in a glossary (see section 4.7)?
- *Unambiguity:* Does the documentation of the requirements allow for only one interpretation or are multiple different interpretations possible? For instance, does a text-based requirement not possess any kind of ambiguity?
- *Conformity to documentation rules:* Have the predetermined documentation rules and documentation guidelines been met? For instance, has the syntax of the modeling language been used properly?

7.3.3 Quality Aspect "Agreement"

The quality aspect "agreement" deals with checking requirements for flaws in the agreement of requirements between stakeholders.

Last opportunity for changes

During the course of requirements engineering, stakeholders gain novel knowledge about the system to be developed. Due to this additional knowledge, the opinion of the stakeholders regarding a requirement that has already been agreed upon can change. During requirements validation, stakeholders have the opportunity to requests changes without impairing the subsequent development activities.

Requirements validation with regard to the quality aspect "agreement" is successful when requirements validation has been applied to the following error types and no significant shortcomings have been detected:

Three test criteria of the quality aspect "agreement"

- *Agreed:* Is every requirement agreed upon with all relevant stakeholders?
- *Agreed after changes:* Is every requirement agreed upon with all relevant stakeholders after it has been changed?
- *Conflicts resolved:* Have all known conflicts with regard to the requirements been resolved?

7.4 Principles of Requirements Validation

Considering the following six principles of requirements validation increases the quality of the validation results:

- *Principle 1:* Involvement of the correct stakeholders
- *Principle 2:* Separating the identification and the correction of errors
- *Principle 3:* Validation from different views
- *Principle 4:* Adequate change of documentation type
- *Principle 5:* Construction of development artifacts
- *Principle 6:* Repeated validation

The individual principles are explained in the following sections.

7.4.1 Principle 1: Involvement of the Correct Stakeholders

The choice of stakeholders for requirements validation depends on the goals of the validation as well as the requirements that are to be audited.

When assembling the auditing team, at least the following two aspects ought to be considered.

Independence of the auditor

Generally, it should be avoided that the author of a requirement is also the person to validate it. The author will make use of his or her prior knowledge when reading or reviewing the requirement. This prior knowledge can negatively influence the identification of errors because potential erroneous passages of the requirements documentation or the requirements are implicitly and subconsciously amended by the author's own knowledge and can thus easily be overlooked.

Internal vs. external auditors

Suitable auditors can be identified within or outside of the developing organization. Internal audits are performed by stakeholders that are members of the developing organization and can be used to validate intermediate results or preliminary requirements. An internal validation is easy to coordinate and organize because the stakeholders are available from within the organization. An external audit requires a higher degree of effort because it identifies auditors and (potentially) hires them for payment. In addition, external auditors have to become familiar with the context of the system to be developed. Due to the high effort, an external audit should be performed only on requirements that exhibit a high level of quality.

7.4.2 Principle 2: Separating the Identification and the Correction of Errors

Basic principle

Separation between identifying errors and actually fixing them has proven itself in the domain of software quality assurance. The same principle can be applied to requirements validation. During validation, the flaws identified are documented immediately. After that, each flaw identified is double-checked to determine whether it really is an error.

Concentrating on error identification

Separating error identification and error correction allows auditors to concentrate on the identification. Measures to correct the errors are taken only after identification measures have been completed. This has the advantages that the resources available for error correction can be used purposefully, that premature error identification does not create additional errors, and that no alleged error is fixed that did not need fixing because further investigation of the error may result in the fact that an alleged error is in fact no error at all. That way, potentially present significant errors are less likely to be overlooked because the auditor is concentrating on fixing a previous error instead of identifying new ones.

7.4.3 Principle 3: Validation from Different Views

Perspective-based validation

Validating requirements from different views is another principle that has proven itself in practice. In this principle, requirements are validated and agreed upon from different perspectives (e.g., by different people, see section 7.5.4). Comparable methods are used in other disciplines as well. For instance, in a legal trial, circumstances are often reported from the perspective of different people so that a sound overall picture can be gained.

7.4.4 Principle 4: Adequate Change of Documentation Type

Strengths and weaknesses of documentation types

Changing the documentation type during requirements validation uses the strengths of one documentation type to balance out the weaknesses of other documentation types. For instance, good understandability and expressiveness are strengths of natural language texts. However, their weakness is potential ambiguity and difficulty in expressing complex circumstances. Graphic models are able to depict complex circumstances rather well, but the individual modeling constructs are restricted in expressiveness.

Simpler identification of errors

Transcribing a requirement that is already documented in another form of documentation simplifies finding errors. For instance, ambiguities in natural language requirements can be identified much easier by transcribing them into a model-based representation.

7.4.5 Principle 5: Construction of Development Artifacts

Suitability of the requirements for design, test, and manual creation

Constructing development artifacts aims at validating the quality of requirements that are meant to be the basis of creating design artifacts, test artifacts, or the user manual. During the course of the validation, the activities usually carried out during subsequent phases to construct respective development artifacts are carried out for small samples. For instance, the auditor intensively deals with a requirement by creating a test case. This way, errors (e.g., ambiguity) can be identified in the requirement. This kind of validation, however, demands a lot of resources because subsequent development activities must be executed at least in part.

7.4.6 Principle 6: Repeated Validation

Validation occurs at a distinct point in time during the development process and relies on the level of knowledge of the auditors at that point in time. During requirements engineering, the stakeholders gain additional knowledge about the planned system. Therefore, a positive validation of requirements does not guarantee that requirements are still valid at a later point in time. Requirements validation should occur multiple times in the following cases (among others):

- Lots of innovative ideas and technology used in the system
- Significant gain of knowledge during requirements engineering
- Long-lasting projects

- Very early requirements validation
- Unknown domain
- Requirements that are to be reused

7.5 Requirements Validation Techniques

In the following sections, techniques for requirements validation are introduced. Often, manual validation techniques, which are also known by the general term *review*, are used for requirements validation. Three major types of reviews can be differentiated:

- Commenting
- Inspections
- Walk-throughs

Along with reviews, the following three techniques have proven themselves to be useful for requirements validation:

- Perspective-based reading
- Validation through prototypes
- Using checklists for validation

In the following, these six techniques are described. Prior to applying any of these techniques, preparatory steps need to be taken as needed, such as identifying and inviting the right stakeholders or organizing suitable rooms and supplies.

7.5.1 Commenting

Individual validation of requirements

During commenting, the author hands his or her requirements over to another person (e.g., a co-worker). The goal is to receive the co-worker's expert opinion with regard to the quality of a requirement. The co-worker reviews the requirement with the goal to identify issues that impair requirement quality (e.g., ambiguity or errors) with respect to predetermined quality criteria. The identified flaws are marked in the requirements document and briefly explained.

7.5.2 Inspection

Typical phases of an inspection

Inspections of software or any other type of product are done to systematically check development artifacts for errors by applying a strict process [Laitenberger and DeBaud 2000].

An inspection is typically separated into various phases [Gilb and Graham 1993]: planning, overview, defect detection, defect correction, follow-up, and reflection. For requirements validation, the planning, overview, error detection, and error collection phases are relevant (see principle 2, separating the identification and correction of errors in section 7.4.2). Individual preparation is an obligatory part of inspections. An inspection session usually serves the purpose of collecting and evaluating error indications. Occasionally, performing dedicated inspection sessions is omitted when performing inspections.

Planning

Among other things, the goal of the inspection, the work results that are to be inspected, and the roles and participants are determined during the planning phase.

Overview

In the overview phase, the author explains the requirements to be inspected to all team members so that there is a common understanding about the requirement among all inspectors.

Error detection

In the error detection phase, the inspectors search through the requirement for errors. Error detection can be performed individually by each inspector or collaboratively in a team. Individual inspection has the advantage that each inspector can concentrate on the requirements. On the other hand, team inspections have the advantage that communication between the inspectors creates synergy effects during error detection. During the course of error detection, any errors that are found are purposively documented.

Error collection and consolidation

In the error collection phase, all identified errors are collected, consolidated, and documented. During consolidation, errors that have been identified multiple times or errors that aren't really errors are identified. The latter can be the case if, for instance, an inspector makes wrong assumptions about a requirement or interprets some constraint the wrong way. Along with consolidation, the identified errors and correcting measures are documented in an error list. Inspections are also known as *technical reviews*.

Roles during inspection

For an inspection to be performed, the following roles must be staffed with suitable personnel:

- *Organizer:* The organizer plans and supervises the inspection process.
- *Moderator:* The moderator leads the session and ensures that the predetermined inspection process is followed. It is advisable to select a neutral moderator because the moderator could potentially balance out opposing opinions of authors and inspectors.
- *Author:* The author explains the requirements that he created to the inspectors in the overview phase and later on is responsible for correcting the errors identified.
- *Reader:* The reader introduces the requirements to be inspected successively and guides the inspectors through them. The role of the reader should be given to a neutral stakeholder so that the inspectors can center their attention on the requirements instead of on the interpretation of the author. Often, the moderator is also the reader.
- *Inspectors:* The inspectors are responsible for finding errors and communicating their findings to the other members of the project team.
- *Minute-taker:* This person takes minutes of the results of the inspection.

7.5.3 Walk-Through

Lightweight inspection

In requirements validation, a walk-through is a lightweight version of an inspection. A walk-through is less strict than an inspection and the involved roles are differentiated to a lesser degree. During a walk-through, at least the roles of the reviewer (comparable to the inspector), author, and minute-taker, and potentially the moderator, are staffed.

Discussion of the identified flaws in quality during a group session

The goal of a walk-through of requirements is to identify quality flaws within requirements by means of a shared process and to gain a shared understanding of the requirements between all the people involved. To prepare for a walk-through, the requirements to be validated are handed out to all participants and inspected. During the walk-through session, the participants discuss the requirements to be validated step-by-step, under guidance of the moderator/reader. Usually, the author of a requirement is the one who introduces the requirement to all other participants. This way, the authors have the opportunity to give additional information to the group along with the actual requirement (e.g., alternative requirements, decisions, and rationale for decisions). A minute-taker documents the flaws in quality that have been identified during the session.

7.5.4 Perspective-Based Reading

Check requirements from a defined perspective.

Perspective-based reading is a technique for requirements validation in which requirements are checked by adopting different perspectives [Basili et al. 1996]. Typically, perspective-based reading is applied in conjunction with other review techniques (e.g., during inspections or walk-throughs). Focusing on particular perspectives when reading a document verifiably leads to improved results during requirements validation. Possible perspectives for validation, for instance, emerge from the different addressees of a requirement [Shull et al. 2000]:

- *User/customer perspective:* The requirements are checked from the perspective of the customer or the user to determine whether they describe the desired functions and qualities of the system.
- *Software architect perspective:* The requirements are checked from the perspective of the software architect to ascertain if they contain all necessary information for architectural design (e.g., if all relevant performance properties have been described).
- *Tester perspective:* The requirements are checked from the perspective of the tester to establish whether they contain the information necessary to derive test cases from the requirements.

Perspective quality aspects

The three quality aspects (see section 7.3) also describe three possible perspectives for requirements validation:

- *Content perspective:* With the content perspective, the auditor verifies the content of requirements and focuses on the quality of the content of the documented requirement.
- *Documentation perspective:* With the documentation perspective, the auditor ensures that all documentation guidelines for requirements and requirements documents have been met.
- *Agreement perspective:* With the agreement perspective, the auditor checks if all stakeholders agree on a requirement, i.e., if the requirements are agreed upon and conflicts have been resolved.

In addition, further perspectives that emerge from the individual context of the development project can be created as need be.

Define validation directives for each perspective.

During perspective-based validation, each auditor is assigned a perspective (at the proper point in time) from which she reads and validates the requirement. For each perspective defined, detailed instructions for performing the validation should be laid down because the auditor might

not be familiar with all relevant details of her assigned perspective. It is advisable to associate questions with each validation instruction that must be answered by the content of the requirements or by the auditor after she has read the requirement, respectively. In addition, validation instructions can be amended with a checklist that summarizes the most important content aspects that ought to be addressed by a requirement with regard to the appropriate perspective.

Follow-up

During the course of the follow-up to a perspective-based reading session, the results of the chosen perspective are analyzed and consolidated. On the one hand, the results of the perspective-based reading contain answers to the predefined questions, and on the other hand, open issues that the auditors noticed while reading may be present. The consolidation can be done as a group effort, similarly to a review.

Support of other techniques

Perspective-based reading can be both an independent technique for requirements validation and a support technique for other validation techniques, such as inspections or reviews of requirements documents by means of perspective-based reading.

7.5.5 Validation through Prototypes

Requirements validation by means of prototypes allows auditors to experience the requirements and to try them out. Experiencing requirements directly through prototypes [Jones 1998] is the most effective method to identify errors in requirements. Stakeholders can try out the prototype and compare their own idea of how the system ought to be implemented with the prototype at hand and thereby find discrepancies between their ideas and the current implementation.

Evolutionary vs. throw-away prototypes

Depending on the further use of the prototype, one can distinguish between throw-away prototypes and evolutionary prototypes [Sommerville 2007]. Throw-away prototypes are not maintained once they have been used. Evolutionary prototypes are developed with the goal to be developed further and improved in later steps. In contrast to throw-away prototypes, implementation plays a much more significant role here. Therefore, the effort to create evolutionary prototypes is much higher.

Selection of relevant requirements

Before a prototype can be implemented, the requirements that shall be validated through the prototype must be selected. The set of requirements to be validated is limited by development resources (e.g., time, money, etc.) that can be allocated for validation. For example, a selection criterion can be the criticality of a requirement.

Preparation of the validation

The following preparations have to be made in order to validate requirements by means of prototypes:

- *Manual/instructions:* The users of the prototype must be supplied with the necessary information so that they can use or apply the prototype. This can be done by means of a manual or by means of proper instruction.
- *Validation scenarios:* Validation scenarios that the users of the prototype can perform with the prototype should be prepared. A validation scenario defines, for example, all relevant data sets or user interactions.
- *Checklist with validation criteria:* For requirements validation, a checklist of validation criteria should be created according to which the prototype (and by proxy, the requirements) can be validated.

Performing the validation

The auditor should validate the prototype without being influenced; i.e., the auditor should execute the validation scenarios independently and by herself. This ensures that the validation results are created without bias.

During validation, the auditors can and ought to execute alternative and deviant scenarios and should use the prototype exploratively and experimentally once the required validation scenarios have been covered. For example, error cases that have remained hidden until then can be identified. For experimental validation of the prototype, the auditor needs to know the scope of the prototype, i.e., the set of requirements that have been considered when the prototype was created. Without knowledge of the implemented requirements, an auditor cannot decide whether an identified error can be traced back to a missing requirement or if the requirement has been consciously omitted in the prototype.

Documentation of the validation results

Requirements validation through prototypes therefore permits two types of result documentation:

- *Protocol of the auditor:* The auditor documents the results and experiences made during the validation of the prototype, e.g., by means of validation scenarios as well as a checklist that he has been supplied with.
- *Observation protocol:* The auditor can be observed by a second person. The second person creates a so-called observation protocol. This protocol can disclose additional important symptoms for errors in requirements. For example, when the auditor hesitates at a certain step in the validation scenario while using the prototype and the observer

documents this, it may be an indication for missing apparentness and as such an indication for impaired understandability of the prototype. Under certain circumstances, it may be advisable to record the validation on video because the validation situation can be analyzed in detail during the follow-up. For example, a video recording can show the realization of requirements pertaining to anthropometric properties (such as ergonomics) or intuitive use and can be investigated in detail.

Analysis

The results of the validation are analyzed after validation is complete. Change suggestions for the requirements are consolidated. If significant changes to the requirements are necessary, it may be advisable to revise the prototype and validate anew.

7.5.6 Using Checklists for Validation

A checklist comprises a set of questions and/or statements about a certain circumstance. Checklists can be applied whenever many aspects must be considered in a complex environment and no aspect must be omitted. A checklist for requirements validation contains questions that ease the detection of errors [Boehm 1984]. Using checklists for requirements validation is very common in practice. Checklists can be used in all previously introduced techniques for requirements validation.

Creating checklists

Before a checklist can be used, every single question or statement must be defined. The sources for questions and statements in the following list can be used to create checklists to support requirements validation:

- The three quality aspects of requirements (see section 7.3)
- Principles of requirements validation (see section 7.4)
- Quality criteria for requirements documents (see section 4.5)
- Quality criteria for individual requirements (see section 4.6)
- Experiences of the auditors from prior projects
- Error statistics [Chernak 1996]

Improving checklists

Checklists are not necessarily complete. When using a checklist, one should always look for opportunities to improve the checklist for future use. For example, if a question was forgotten, the checklist should be amended to contain the extra question. Ambiguous questions or questions that are not understandable must be marked and revised. Outdated or no longer valid questions should be removed.

Checklists as a guideline

Checklists can support requirements validation in different ways. They can serve as a guideline for the auditor, who can follow the checklists at her own discretion (e.g., during a review).

Checklists as a means of structuring

The checklist can define a list of questions that must be strictly adhered to. These questions must be answered by the auditor to validate the requirements. In this case, the checklist serves as a measure to approach the validation in a structured manner. For example, the checklist may detail the exact process that the auditors are asked to apply, which guarantees that every auditor validates the requirements in the same way. This makes the results more comparable.

Hybrid forms of checklist application are also possible. For example, a checklist can contain obligatory questions for perspective-based reading and can contain suggestions that the auditor may or may not follow.

Successfully applying checklists

Applying checklists for requirements validation successfully depends on the manageability and complexity of the checklist. A large amount of questions can make it more difficult to use the checklist because the auditor does not have a steady overview of the questions and is thus forced to consult the checklist frequently. It is therefore advisable to design the checklist to not be longer than a single page [Gilb and Graham 1993]. In addition, questions that are formulated altogether too generically or abstractly can make it more difficult to use the checklist. For example, the question "Is the requirement formulated appropriately?" can lead to a multitude of different answers, depending on what the auditor considers an appropriately formulated requirement. The questions therefore ought to be as precise as possible.

7.6 Requirements Negotiation

To negotiate the requirements of a system to be developed, it is necessary to identify conflicts and to resolve those conflicts. This is done by means of systematic conflict management. The conflict management in requirements engineering comprises the following four tasks:

Four tasks of conflict management

- Conflict identification
- Conflict analysis
- Conflict resolution
- Documentation of the conflict resolution

These four tasks are explained in the following sections.

7.6.1 Conflict Identification

Conflicts can arise during all requirements engineering activities. For example, different stakeholders can utter contradicting requirements during elicitation.

Conflict identification in all requirements engineering activities

Conflicts between requirements and conflicts between stakeholders are often not obvious due to different reasons. During the entire requirements engineering process, the requirements engineer should pay attention to potential conflicts so that they can be identified, analyzed, and resolved early on.

7.6.2 Conflict Analysis

Determining the conflict type

During conflict analysis, the reason for an identified conflict must be determined. According to [Moore 2003], different types of conflicts exist.

Data conflict

A *data conflict* between two or more stakeholders is characterized by a deficit of information, by false information, or by different interpretations of some information. For example, take the following requirement: "R131: The reaction time of the planned system shall not exceed one second". A data conflict between two stakeholders with regard to this requirement can arise from the fact that one stakeholder considers a reaction time of 1 second to be too slow while another stakeholder does not believe that a reaction time of 1 second is feasibly implementable (i.e., it is too short).

Conflict of interest

A *conflict of interest* between two or more stakeholders is characterized by subjectively or objectively different interests or goals of stakeholders. A conflict of interest between two or more stakeholders can arise, for instance, when one stakeholder primarily focuses on keeping the costs of the planned system at a minimum while another stakeholder primarily desires a high level of quality. A conflict of interest between these two stakeholders arises when the first stakeholder rejects a requirement due to estimated costs and the second stakeholder insists on implementing it due to quality reasons.

Conflict of value

A *conflict of value* is characterized by differing underlying values stakeholders have regarding some circumstance (e.g., cultural differences, personal ideals). For instance, a conflict of value arises when one stakeholder favors open source technologies while another stakeholder favors closed sources technologies.

Relationship conflict

A *relationship conflict* is characterized by strong emotions, stereotypical relationship concepts, deficient communication, or negative inter-

personal behavior between stakeholders (e.g., insults, disrespect). For instance, a relationship conflict arises when two stakeholders of equal rank or position (e.g., department leaders) reject each other's requirements and try to distinguish themselves by forcing their requirements onto the project.

Structural conflict

A *structural conflict* is characterized by unequal levels of authority or power. For instance, a structural conflict can arise between an employee and his superior if the superior invariably rejects requirements that the employee has defined because he does not recognize the employee's competence to delineate requirements.

Mixed reasons for conflicts

Often, it is difficult to unambiguously classify emerging conflicts. For example, a conflict can be a relationship conflict with clear structural components. Similarly, a conflict of interest can be a conflict of values as well. Therefore, it is advisable to analyze an identified conflict with respect to all types so that all possible reasons for the conflict can be determined and suitable resolution strategies can be selected.

7.6.3 Conflict Resolution

Good conflict resolution is a success factor.

Conflict resolution is very important for requirements negotiation because the strategy of conflict resolution has a big influence on the willingness of the people involved (e.g., customers, consultants, or developers) to continue working together. For example, a conflict resolution considered unfair by at least one party of the conflict can lead to a decreased engagement and willingness to collaborate in the project. On the other hand, a resolution that is considered fair by all parties can increase the willingness to cooperate because this signals that everyone's ideas about the planned system are being considered.

Involvement of the relevant stakeholders

Independently from the selected resolution strategy, it is essential to involve all relevant stakeholders. If not all relevant stakeholders are considered, some opinions and viewpoints will remain unconsidered. The conflict will therefore only be resolved in part or incompletely. In the following paragraphs, different conflict resolution techniques are introduced.

Agreement

With the conflict resolution technique *agreement*, all conflict parties negotiate a solution to the conflict. The parties exchange information, arguments, and opinions and try to convince one another of each other's viewpoints in order to come to an agreeable solution.

Compromise

With the conflict resolution technique *compromise*, all conflict parties try to find a compromise between alternative solutions. In contrast to an

agreement, a compromise consists of an amalgamation of different parts of the alternative solutions. Also, a compromise can mean that all alternative solutions as proposed thus far are discarded and entirely new solutions are creatively developed.

Voting

In the conflict resolution technique *voting*, all conflict parties vote on solution alternatives. The alternatives that are up for voting are presented to all relevant stakeholders. Each stakeholder casts her vote for an alternative and the alternative with the most votes is accepted as the resolution for the conflict.

Definition of variants

In the conflict resolution technique *definition of variants*, the system is developed in a way that permits the definition of variants by deriving variants, by selecting parameters that define system variants, or by selecting variable system properties. This way, the system can satisfy the different interests of the stakeholders.

Overruling

In the conflict resolution technique *overruling*, a conflict is resolved by means of the hierarchical organization. This means that a conflict party with higher organizational rank or position wins the conflict by overruling objections of organizationally lower parties. If both parties have the same organizational rank, the conflict is resolved by a superior stakeholder or some third-party decider. This conflict resolution technique is only advisable if other resolution techniques have failed (e.g., no compromise could be found) or are not applicable due to limitations of resources (e.g., time).

Consider-all-facts

In the conflict resolution technique *consider-all-facts (CAF)*, all influencing factors of a conflict are being investigated so that as much information about the conflict can be collected as possible. This information is used during resolution. By prioritizing the influence factors, the relevance is determined. Based on the results of this technique, the plus-minus-interesting conflict resolution technique can be applied.

Plus-minus-interesting

In the conflict resolution technique *plus-minus-interesting (PMI)*, all positive and negative repercussions of a solution alternative are investigated so that positive and negative repercussions can be evaluated. Positive repercussions are placed in the category "plus" and negative repercussions are placed in the category "minus". Repercussions that are neither positive nor negative are placed in the category "interesting". Repercussions in the category "interesting" cannot be evaluated yet and must be investigated further to determine if their influence is positive or negative.

Decision matrix

In the conflict resolution technique *decision matrix*, a table is created that contains solution alternatives in the columns and all relevant decision criteria in the rows. The decision criteria can be identified by means of the

technique "consider-all-facts". For each combination of criterion and solution alternative, an assessment is made, for instance by means of a point-scale ranging from irrelevant (0 points) to relevant (10 points). Table 7-1 shows a decision matrix.

	Solution alternative 1	Solution alternative 2	Solution alternative 3
Criterion 1	3	6	2
Criterion 2	5	4	10
Criterion 3	10	3	5
Sum	18	13	17

Table 7-1 *Decision matrix*

In order to find a solution, the sums of the columns are calculated; i.e., the assessments of the criteria of each solution alternative are summed up. The solution alternative with the highest score is accepted as the decision. In the example shown in table 7-1, this would be solution alternative 1.

7.6.4 Documentation of the Conflict Resolution

Risks of missing conflict documentation

Conflicts cannot be avoided during requirements engineering. A resolution to a conflict must always be traceably documented. If a conflict resolution is not properly documented, the following threats (among others) to the project may arise:

- *Handling conflicts repeatedly:* A certain conflict can arise a second time during the requirements engineering process. Without proper documentation of the conflict resolution, the conflict must be analyzed and resolved anew. This causes additional effort and can potentially lead to additional conflicts or abrogate previous resolutions.
- *Inappropriate conflict resolution:* During the requirements engineering process, the resolution of a conflict can turn out to be wrong or unsuitable. In this case, the conflict must be investigated and resolved anew. Without proper documentation, relevant information that has been considered during the initial analysis and resolution can be overlooked and the new conflict resolution can once again lead to false results.

In both cases, proper documentation of the conflict and its resolution supports the requirements engineering process and ensures that relevant information already known can be considered.

7.7 Summary

The quality of the elicited and documented requirements must be assured during requirements engineering so that it can be guaranteed that the requirements meet the desires and ideas of the stakeholders adequately. Therefore, it is necessary to validate the requirements with regard to the quality of their content, their documentation, and their agreement with respect to the different stakeholders. There are different techniques that can be selected and purposively combined for requirements validation, depending on the project peculiarities and project goals. Among the most common validation techniques for requirements are the different types of requirements reviews (e.g., commenting, inspection, walk-through) as well as perspective-based reading and validation through prototypes and checklists.

For requirements negotiation, it is necessary to identify conflicts between stakeholders, analyze them, and resolve them in a suitable manner. A systematic conflict management supports analysis and resolution of the conflicts that have been identified over the course of requirements validation or other requirements engineering activities.

This page intentionally left blank

8 Requirements Management

Requirements management comprises purposefully assigning attributes to requirements, defining views on requirements, prioritizing requirements, and tracing requirements as well as versioning requirements, managing requirements changes, and measuring requirements. Requirements management includes the management of individual requirements as well as the management of requirements documents.

8.1 Assigning Attributes to Requirements

Information about the requirements must be documented throughout the entire life cycle of a system. This includes, for example, unique identifiers of a requirement, the name of the requirement, the author and sources of the requirement, and the person responsible for the requirement.

8.1.1 Attributes for Natural Language Requirements and Models

To document information about requirements, it has proven useful to delineate this information in a structured manner: as attributes. Attributes of a requirement are defined by a unique name, a short description of the contents, and the set of possible values that can be assigned to the attribute.

Template-based assignment of attributes to requirements

The simplest way to define requirement attributes is by means of a table structure (template). The template defines the relevant information that is to be documented. This information, i.e., the defined attributes (attribute types), can be different for each type of requirement. For example, the template for functional requirements can be different from the template for quality requirements with respect to the defined attribute types and/or the allowed attribute values.

8.1.2 Attribute Scheme

The set of all defined attributes for a class of requirements (e.g., functional requirements, quality requirements) is called an attribute scheme. Attribute schemes are usually tailored to meet the individual project's needs.

Assignment of requirement attributes

During the course of the project, the attributes of the requirements are assigned with fitting attribute values. Figure 8-1 shows an exemplary assignment of attributes for a requirement including the attributes "identifier", "name", and "requirement description" as well as attributes that allow for documenting the stability of the requirements and its source as ***well as its author.***

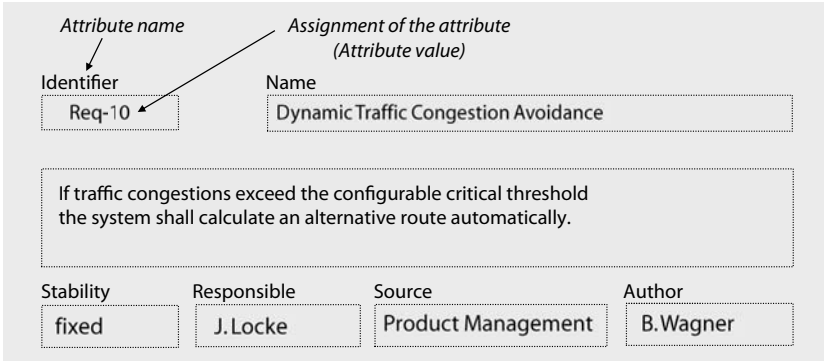

Figure 8-1 *Example of requirement attribute assignment*

The requirement that is documented on the basis of the simple template shown in figure 8-1 has the code "Req-10" as its unique identifier. It bears the name "Dynamic Traffic Congestion Avoidance" and a description that specifies the subject of this requirement. The stability of this requirement is classified as "fixed", "J. Locke" is the person responsible for this requirement, and the requirement stems from the source "Product Management". "B. Wagner" is the author.

The reader of the requirement (e.g., the contractor, product manager, developer, project manager) has a significant advantage when template-based documentation is used, namely that information of the same type can always be found in the same position (e.g., the requirement stability is always in the template section "stability"). In addition, template-based assignment of attributes has the advantage for the requirements engineer that it is harder for her to overlook important information and that this information, supported by the structure of the template and the

predetermined attribute values, can be documented purposefully and correctly.

8.1.3 Attribute Types of Requirements

Frequently used attribute types

The various standards in requirements engineering and the most pertinent tools for requirements documentation and management often offer a set of predefined attributes. Table 8-1 lists attribute types that are frequently used in practice during requirements management.

Attribute Type	Meaning
Identifier	Short, unique identifier of a requirement artifact from the set of all regarded requirements.
Name	Unique, characterizing name.
Description	Briefly describes the content of the requirement.
Version	Current version of the requirement.
Author	Specifies the author of the requirement.
Source	Specifies the source or sources of the requirement.
Stability	Specifies the approximate stability of the requirement. The stability is the amount of changes that are to be expected with regard to the requirement. Possible values can be "fixed", "established", and "volatile".
Risk	Specifies the risk based on an estimate of the amount of damage and loss and the probability of occurrence.
Priority	Specifies the priority of the requirement regarding the chosen prioritization properties, e.g., "importance for market acceptance", "order of implementation", "loss/opportunity cost if not implemented".

Table 8-1 *Frequently used attribute types*

Additional attribute types for requirements

Along with the requirements attributes listed in table 8-1, many additional attribute types exist to document important information of a requirement. Table 8-2 shows a selection of additional attribute types for requirements.

Attribute Type	Meaning
Person responsible	Specifies the person, group of stakeholders, organization, or organizational unit that is responsible for the content of the requirement.
Requirement type	Specifies the type of requirement (e.g., "functional requirement", "quality requirement", or "constraint") depending on the applied classification scheme.
Status regarding the content	Specifies the current status of the content of the requirement, e.g., "idea", "concept", "detailed content".
Status regarding the validation	Specifies the current status of the validation, e.g., "unvalidated", "erroneous", "in correction".
Status regarding the agreement	Specifies the current status of the negotiation, e.g., "not negotiated", "negotiated", "conflicting".
Effort	Estimated/actual effort to implement the requirement
Release	The designation of the release in which the requirement shall be implemented.
Legal obligation	Specifies the degree of legal obligation of the requirement.
Cross references	Specifies relations to other requirements, for example, if it is known that the implementation of the requirement requires prior implementation of another requirement.
General information	In this attribute, arbitrary information that is considered relevant can be documented, for example, if the negotiation of this requirement is scheduled for discussion during the next meeting with the stakeholders.

Table 8-2 *Additional types of requirement attributes*

Project-specific tailoring of the attribute scheme

The attribute types suggested are the basis for defining an attribute scheme in the development project. To define an attribute scheme, at least the following specific aspects must be considered:

- Specific properties of the project, e.g., project size, local or distributed development, or project risk
- Constraints of the organization, e.g., organizational standards and regulations
- Properties and regulations of the application domain, e.g., reference models, modeling guidelines, standards
- Constraints and restrictions of the development process, e.g., liability law, process standards

Definition of attributes by means of information models

When employing tools for requirements management, defining the attribute structure of requirements is often not done by means of tables, but is model based, by means of information models. A model-based definition of an attribute scheme determines the attribute types as well as limitations in attribute values, similar to template-based definitions. In addition, model-based attribute scheme definition allows for determining relations between attribute types of different attribute schemes.

Advantages of model-based attributing

Along with the advantages of table-based definition, model-based assignment of requirement attributes additionally allows consideration of requirement dependencies when selectively accessing the requirements. This aids in maintaining consistency in the attributes of the requirements. Furthermore, the information model of a model-based assignment of requirement attributes can serve as the foundation for the definition of an attribute structure to be used in a requirements management tool. (see section 9.3). Also, templates for the assignment of requirement attributes can be generated on the basis of the information model.

8.2 Views on Requirements

Structuring requirements by means of information models allows for generating specific views on requirements. It can be seen in practice that the amount of requirements and the amount of dependencies among requirements are evermore increasing. In order to keep the complexity of the requirements manageable for the project staff, it is necessary to selectively access and thereby filter the requirements depending on the current task.

Role-specific definition of views

Views on requirements are often defined for different roles in the development process. Examples include views for the architect, the programmer, the project manager, and the tester. It is common to define multiple views for a role in order to support the sub-activities of each role. One particular view can also be applied to multiple roles.

8.2.1 Selective Views on the Requirements

A view contains a part of all available requirement information. A view can do the following:

- Select particular requirements; i.e., not every requirement is contained in a view.

- Mask certain attributes of requirements; i.e., not every attribute of a requirement is contained in a view.
- Arbitrarily combine both these selection principles; i.e., only a subset of all available requirements and only a subset of all available attributes are contained in a view.

Generating selective views

Figure 8-2 illustrates the generation of three views, represented by a table that is defined on the basis of the structure of the attributes. In all three cases, the views are created by selecting attribute types as well as by determining the attributes that must be available. The definition of the first view (❶), for example, determines that only those requirements are selected that "J. Locke" is responsible for and that have a stability of "fixed". Of all selected requirements, only the attributes "identifier", "name", "description", and "author" are being considered.

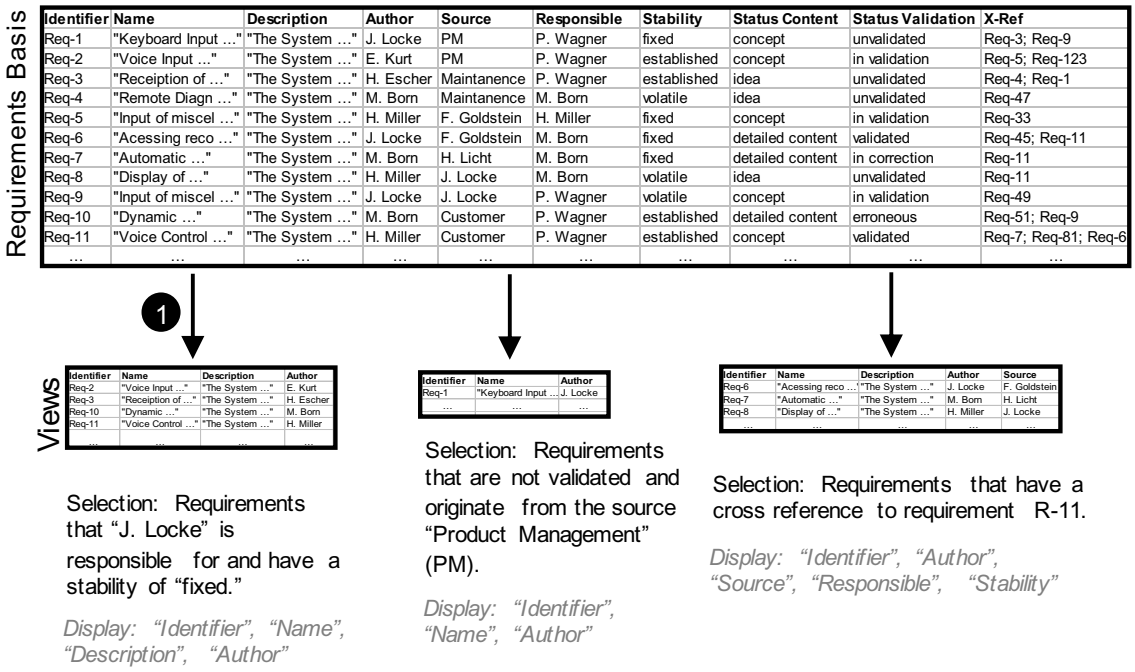

Figure 8-2 *Selective views on the requirements*

8.2.2 Condensed Views on the Requirements

Along with selecting existing information from the requirements basis, views can contain generated or condensed data that is not immediately contained in the requirements. Views that contain only generated or condensed data are called condensed views.

Generating condensed views

Condensed views can be defined by aggregating the data contained in the requirements basis. A condensed view can, for example, contain information on the percentage of requirements that stem from a particular source.

Combination of selecting and condensing

A single view may also consist of a combination of generated, condensed, and selected data.

Requirements Basis

Identifier	Name	Description	Author	Source	Responsible	Stability	Status Content	Status Validation	X-Ref
Req-1	Keyboard Input ...	The System ...	J. Locke	PM	P. Wagner	fixed	concept	unvalidated	Req-3; Req-9
Req-2	Voice Input ...	The System ...	E. Kurt	PM	P. Wagner	established	concept	in validation	Req-5; Req-123
Req-3	Receiption of ...	The System ...	H. Escher	Maintanence	P. Wagner	established	dea	unvalidated	Req-4; Req-1
Req-4	Remote Diagn ...	The System ...	M. Born	Maintanence	M. Born	volatile	dea	unvalidated	Req-47
Req-5	Input of miscel ...	The System ...	H. Miller	F. Goldstein	H. Miller	fixed	concept	in validation	Req-33
Req-6	Acessing reco ...	The System ...	J. Locke	F. Goldstein	M. Born	fixed	detailed content	validated	Req-45; Req-11
Req-7	Automatic ...	The System ...	M. Born	H. Licht	M. Born	fixed	detailed content	in correction	Req-11
Req-8	Display of ...	The System ...	H. Miller	J. Locke	M. Born	volatile	dea	unvalidated	Req-11
Req-9	Input of miscel ...	The System ...	J. Locke	J. Locke	P. Wagner	volatile	concept	in validation	Req-49
Req-10	Dynamic ...	The System ...	M. Born	Customer	P. Wagner	established	detailed content	erroneous	Req-51; Req-9
Req-11	Voice Control ...	The System ...	H. Miller	Customer	P. Wagner	established	concept	validated	Req-7; Req-81; Req-6
...	...	...	...	...	...	...	...	...	...

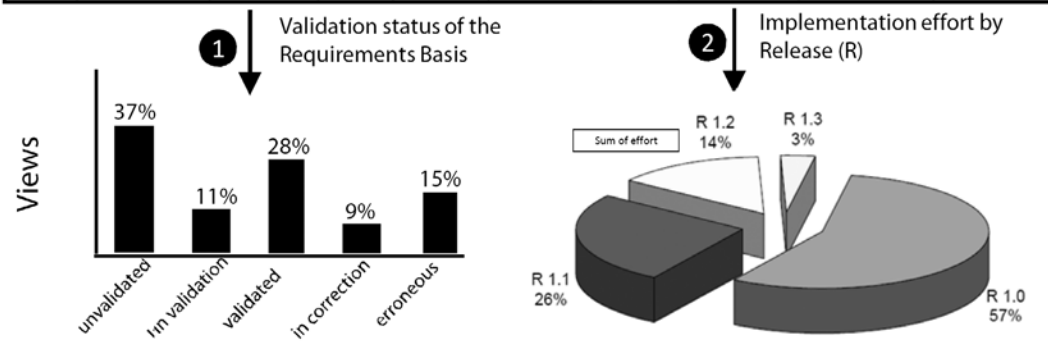

Figure 8-3 *Condensed view generated from a requirements basis*

Figure 8-3 shows two condensed views of the requirements. The view "Validation status of the Requirements Basis" (❶) groups requirements according to the current status of validation and calculates the percentage value of the requirements with regard to the status "unvalidated", "in validation", "validated", "in correction", and "erroneous". The result is depicted as a bar chart in the figure above. In view (❷), "Implementation effort by Release", the estimated and actual effort involved with the implementation

of the requirements of a particular release is depicted. In order to calculate this aggregated data, the requirements are grouped by their respective release and their implementation effort is summed up. The result is depicted as a pie chart in figure 8-3.

8.3 Prioritizing Requirements

Requirements are prioritized during requirements engineering using different prioritization criteria in all sub-activities. Requirements can be prioritized by their order of implementation, for example. Due to the different prioritizations in the various sub-activities, the priority of a requirement can be determined by one or more attributes (e.g., priority of the contractor, priority due to urgency of implementation).

8.3.1 Method for Requirements Prioritization

Determining goal and constraints of prioritization

In order to prioritize a set of requirements, a goal (i.e., purpose) of prioritization must be defined first. In addition, the constraints of prioritization are documented, such as the availability of different stakeholders and groups thereof or the resources available for prioritization.

Determining prioritization criteria

Depending on the goal of prioritization, the criterion for prioritizing the requirements (or the combination of two or more criteria) is chosen. The following are typical examples of prioritization criteria:

- Cost of implementation
- Risk
- Damage due to unsuccessful implementation
- Volatility
- Importance
- Duration of implementation (i.e., how long it takes to be implemented)

Determining Stakeholders

Depending on the goal of prioritization and the selected prioritization criteria, it is usually necessary to involve different stakeholders in the prioritization process. By choosing appropriate stakeholders, it can be guaranteed that the required expert knowledge is available during the prioritization process. The stakeholders that ought to be involved are, depending on the goal and prioritization criteria, developers, project managers, customers, or users, for example.

Selection of artifacts

In addition, the requirements to be prioritized must be selected. When selecting requirements, one must make sure that the selected requirements stem from the same level of abstraction. Prioritizing requirements from considerably differing levels of detail will lead to inconsistent and erroneous results because stakeholders tend to assign a higher priority to requirements at higher levels of abstraction than to more refined and concrete requirements.

Selection of prioritization techniques

On the basis of the determined properties of the prioritization (e.g., constraints, criteria of prioritization, etc.), a suitable prioritization technique or a combination of multiple techniques is selected.

8.3.2 Techniques for Requirements Prioritization

For prioritization, multiple techniques exist. The techniques mainly differ with regard to the time and effort needed but also with regard to the suitability of the different prioritization criteria and project properties.

Ad hoc techniques and analytical techniques

The spectrum of prioritization techniques spans from simple, single-criterion classification to elaborate analytic prioritization approaches, such as AHP (Analytical Hierarchy Process) [Saaty 1980], Cost-Value-Analysis [Karlsson and Ryan 1997], or QFD (Quality Function Deployment) [Akao 1990].

In many projects, simple ad hoc prioritization techniques such as ranking or requirements classification are well suited. Especially with regard to the resources available, using ad hoc techniques is often advisable.

If the decision process is considered too incomprehensible, or if the results are too erroneous, analytical approaches for prioritization should be used (additionally). In practice, multiple prioritization techniques are used in combination in order to prioritize the requirements [Lehtola and Kauppinen 2006].

Ranking and Top-Ten Technique

Two well-established techniques for requirement prioritization are, for example, the following [Lauesen 2002]:

- *Ranking:* In this technique, a number of selected stakeholders arrange the requirements to be prioritized with respect to a specific criterion.
- *Top-Ten Technique:* In this technique, the *n* most important requirements for a defined criterion are selected. For these requirements, a

ranking order is determined afterward. This ranking order represents the importance of the selected requirements with regard to the defined criterion.

Single-Criterion Classification

Another prioritization technique that is often used in practice is based on the classification of requirements with respect to the importance of the realization of the requirements for the system's success. This type of prioritization is based on assigning each requirement to one of the following priority classes [IEEE 830-1998]:

- *Mandatory:* A mandatory requirement is a requirement that must be implemented at all costs or else the success of the system is threatened.
- *Optional:* An optional requirement is a requirement that does not necessarily need to be implemented. Neglecting a few requirements of this class does not threaten the success of the system.
- *Nice-to-have:* Nice-to-have requirements are requirements that do not influence the system's success if they are not implemented.

In practice, differentiating between "optional" and "nice-to-have" requirements can be very difficult. Therefore, requirements classification demands classification criteria that are as objectively verifiable as possible.

Kano Classification

The Kano approach introduced in section 3.2 also supports the prioritization of requirements. By making use of the Kano approach, one can classify and prioritize requirements with respect to their acceptance on the market. In order to do so, the following three property classes (see also figure 3-1) are classified:

The three properties in the Kano approach

- *Dissatisfiers:* A requirement specifies a dissatisfier the system must possess in order to be successfully introduced to the market.
- *Satisfiers:* A requirement specifies a satisfier if the customers consciously demand the associated property. Satisfiers of the system specify the degree of satisfaction of the customer. An increase in the number of satisfiers usually leads to increased customer satisfaction.
- *Delighters:* A requirement specifies a delighter if the customers do not consciously demand the defined system property or the customers do not expect the implementation of the property. The customer satisfaction increases exponentially by implementing delighters.

On the basis of requirements classified according to Kano, a prioritization of the requirements can be performed in order to plan the system releases, for example.

Prioritization Matrix According to Wiegers

Computing requirement priorities

The prioritization matrix according to Wiegers [Wiegers 1999] is an analytical prioritization approach for requirements. The core of the approach is a prioritization matrix according to which the priorities of the regarded requirements can be determined systematically. Figure 8-4 shows the structure of a prioritization matrix according to Wiegers as well as the method according to which priorities are calculated.

Relative weight ❶	→2 (WeightBenefit)	→1 (WeightDetriment)			→1 (WeightCost)		→0.5 (WeightRisk)			
❷ Requirement	Relative Benefit	Relative Detriment	Total	Value %	Relative Cost	Cost %	Relative Risk	Risk %	Priority	Rank
R_1	5	3	13	16.8	2	13,3	1	9,1	0.941	1
R_2	9	7	25	32.5	5	33,3	3	27,2	0.692	3
R_3	5	7	17	22.1	3	20,0	2	18,2	0.759	2
R_4	2	1	5	6.5	1	6,7	1	9,1	0.577	4
R_5	4	9	17	22.1	4	26,7	4	36,4	0.489	5
Total	25	27	77	100	15	100	11	100	—	
	❸	❹	❺		❻		❼		❽	❾

Figure 8-4 *Calculation of priorities in a prioritization matrix according to Wiegers*

Systematic method to determine the requirement priorities

In the following, the calculation of priorities in a prioritization matrix according to Wiegers is only briefly sketched. More detailed information can be found in [Wiegers 1999].

The calculation of priorities in a prioritization matrix according to Wiegers can be done as follows:

❶ Determine the relative weights for benefit, detriment, cost, and risk.
❷ Determine the requirements to be prioritized.
❸ Estimate the relative benefit.
❹ Estimate the relative detriment.
❺ Calculate the total values and percentage values for each requirement:

$$Value\%(R_i) = Benefit(R_i) \times WeightBenefit + Detriment(R_i) \times WeightDetriment$$

❻ Estimate the relative cost and calculate the cost percentage for each requirement.
❼ Estimate the relative risks and calculate the risk percentage for each requirement.
❽ Calculate the individual requirement priorities:

$$Priority(R_i) = Value\%(R_i)/(Cost\%(R_i) \times WeightCost + Risk\%(R_i) \times WeightRisk)$$

❾ Assert the rank of the individual requirements.

It became apparent in practice that analytical prioritization approaches such as the prioritization matrix according to Wiegers as sketched above demand considerably more time and effort than ad hoc approaches, so these ad hoc approaches are to be favored in many cases. However, analytical approaches have the advantage that the degree of subjectivity in the prioritization results can be significantly reduced so that they lead to more objective and comprehensible results in complex and critical prioritization situations.

8.4 Traceability of Requirements

An important aspect of requirements management is ensuring the traceability of requirements. The traceability of a requirement is the ability to trace the requirements over the course of the entire life cycle of the system (see section 4.5.5).

8.4.1 Advantages of Traceable Requirements

Advantages of requirements traceability

The use of traceability information supports system development in many aspects and is often the precondition for establishing and using certain techniques during the developmental process [Pohl 1996; Ramesh 1998]:

- *Verifiability:* Traceability of requirements allows verifying whether a requirement has been implemented in the system, i.e., if the requirement has been implemented through a system property.
- *Identification of gold-plated solutions in the system:* Traceability of requirements allows for the identification of so-called gold-plated solutions of the developed system and thereby allows identifying unneeded properties. In order to do that, for each system property (functional or

qualitative), a check is performed to determine whether it contributes to the implementation of a requirement of the system.

- *Identification of gold-plated solutions in the requirements:* Tracing requirements back to their origin allows identifying requirements that do not contribute to any system goal and are not associated with any source. Usually, there is no reason for these requirements to exist and hence these requirements do not have to be implemented.
- *Impact analysis:* Traceability of requirements allows for the analysis of effects during change management. For example, traceability of requirements allows identifying the requirements artifacts that must be changed when their underlying requirements undergo a change.
- *Reuse:* Traceability of requirements allows for the reuse of requirements artifacts in other projects. By comparing the requirements of a previous project to the requirements of a new project by means of trace links, development artifacts (e.g., components, test cases) can be identified that may be adapted and/or reused in the new development project.
- *Accountability:* Traceability of requirements allows for retroactive assignment of development efforts to a requirement. After the requirement is implemented, for example, all partial efforts for the associated development artifact can be summed up and associated with the requirement.
- *Maintenance:* Traceability of requirements allows for simplified system maintenance. For example, the cause and effect of failures can be identified, the system components that are affected by the failure can be determined, and the effort for removing the underlying error can be estimated.

8.4.2 Purpose-Driven Definition of Traceability

As resources are usually severely restricted during development projects, capturing all conceivable information that supports the traceability of requirements over the course of the system life cycle is almost never possible.

Purpose of traceability information

In order to establish requirements traceability effectively and efficiently, the information to be recorded should be chosen with respect to the purpose that it will serve. In other words, only the information which has a clear purpose for system development or system evolution [Dömges and Pohl 1998; Ramesh and Jarke 2001] ought to be recorded. Recording

of traceability information that is not purpose driven often results in the fact that the recorded information cannot be profitably used in the development project. Traceability information that is recorded in this fashion is often sketchy and incomplete, unstructured, and erroneous with regard to its further use.

8.4.3 Classification of Traceability Relations

Pre-RS traceability and post-RS traceability

The pertinent literature on the topic of requirements traceability suggests different kinds of traceability of requirements. A common differentiation is distinguishing between pre-requirements-specification (pre-RS) traceability and post-requirements-specification (post-RS) traceability of requirements [Gotel and Finkelstein 1994]. We thus distinguish between three kinds of traceability:

- *Pre-RS traceability:* Pre-RS traceability are traceability links between requirements and those artifacts that are the basis for the requirements, e.g., artifacts like the source or origin of a requirement (previous artifacts).
- *Post-RS traceability:* Post-RS traceability comprises traceability information between requirements and artifacts of subsequent development activities. For example, such artifacts could be components, implementation, or test cases that belong to a requirement (posterior artifacts).
- *Traceability between requirements:* The traceability between requirements is about mapping dependencies between requirements. An example of this kind of traceability is the information that a requirement refines another requirement, generalizes it, or replaces it.

Figure 8-5 shows the three types of traceability of requirements in requirements engineering.

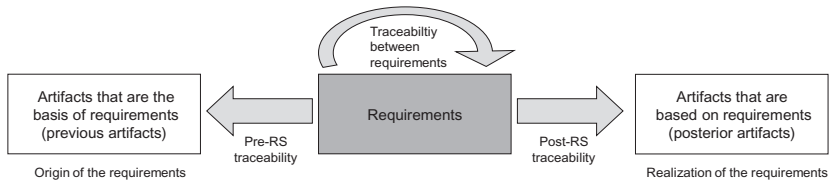

Figure 8-5 *Types of requirements traceability*

Figure 8-6 shows the three types of requirements traceability by means of requirement "R-14" in an example. The pre-RS traceability comprises the relations of requirement "R-14" to its origin. The origin of this requirement are the artifacts in the system context that influence the requirement. The post-RS traceability of requirement "R-14" consists of the relations to the components in the rough design, the refined design, and the respective implementation as well as test cases that are used during system testing and verify the implementation of the requirement in the developed system.

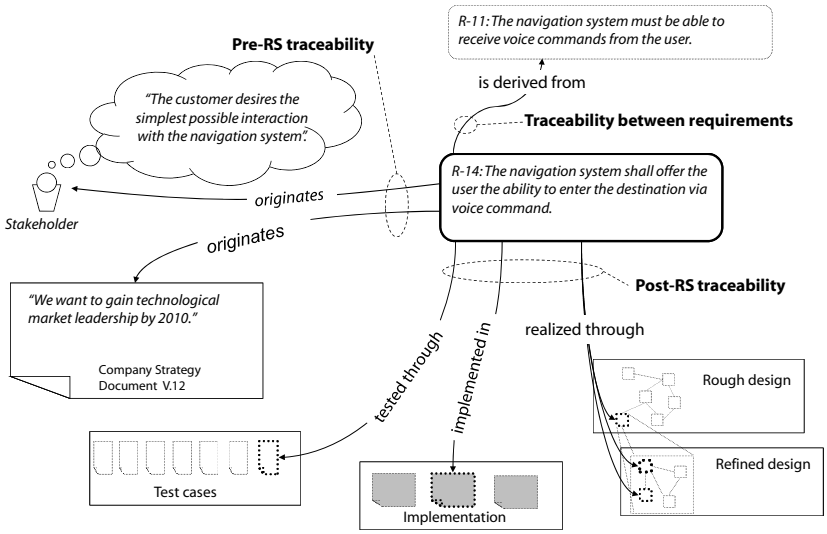

Figure 8-6 *Example of the three types of requirements traceability*

In addition, figure 8-6 shows the traceability between requirements. The traceability link between requirement "R-14" and "R-11" documents that requirement "R-14" was derived from requirement "R-11".

8.4.4 Representation of Requirements Traceability

Requirements traceability information can be represented in different ways. The most common approaches to representing traceability are simple textual references, hyperlinks, and trace matrices and trace graphs.

Text-Based References and Hyperlinks

This simple way to represent traceability information of a requirement consists of annotating the target artifact as a textual reference in the

requirement (initial artifact) or to establish a hyperlink between the initial artifact and the target artifact. When linking artifacts, different types of hyperlinks with specific link semantics can be used.

Trace Matrices

Another common technique for representing and documenting traceability information between requirements as well as between requirements and previous and posterior artifacts in the development process are trace matrices. The rows in a trace matrix contain the initial artifacts (requirements). In the columns, the target artifacts (e.g., sources of requirements, development artifacts, requirements) are represented. If a trace link exists between an initial artifact in row *n* and a target artifact in column *m*, cell *(n, m)* is marked in the trace matrix.

Interpretation of a trace matrix

Figure 8-7 shows a simple trace matrix for the trace relation "derived" that exists between two requirements. An entry in the matrix specifies that a trace link of type "derived" exists from a requirement "*Req-n*" to another requirement "*Req-m*" such that "*Req-n*" was derived from "*Req-m*".

Target artifacts

Initial artifacts	*derived*	Req-1	Req-2	Req-3	Req-4	Req-5
	Req-1		X			
	Req-2			X		
	Req-3					X
	Req-4			X		
	Req-5					

Figure 8-7 *Representation of traceability information in a trace matrix*

Maintainability of trace matrices

In practice, it became apparent that trace matrices are difficult to maintain as the number of requirements increases. A trace matrix that, for example, documents the refinement relations between merely 2,000 requirements contains over four million cells. In addition, many trace matrices must be created in order to be able to represent the available information cleanly (e.g., with regard to different types of traceability links).

Trace Graphs

A trace graph is a graph in which all nodes represent artifacts and all edges represent relationships between artifacts. The distinction between differ-

ent artifacts and types of traceability can be realized by means of assigning different attributes to the nodes and edges of the graph.

Trace graph over different development artifacts.

Figure 8-8 shows the representation of traceability information in a simple example. In the trace graph, a node type is defined for each type of artifact (context information *"C"*, requirements *"Req-n"*, components *"Comp-n"*). In addition, three types of edges are defined to represent three types of traceability relations (*"realized through"*, *"is origin"*, *"refines"*).

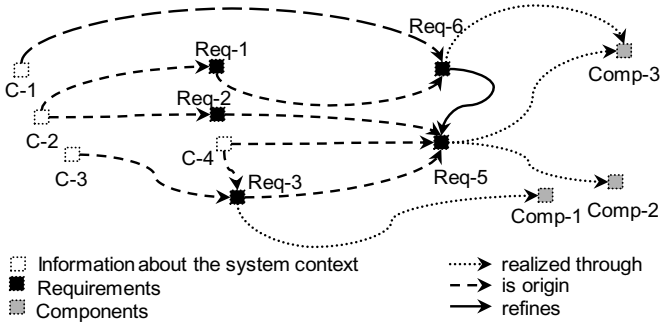

Figure 8-8 *Representation of traceability in a trace graph (extract)*

Traceability chains

If traceability information about previous artifacts (e.g., stakeholders and interview protocols) as well as posterior artifacts (e.g., test cases and components) must be managed, traceability chains for the respective requirement can be created at different levels, up to a trace of the requirement over the entire life cycle of the system. Common tools to maintain requirements allow for the definition of representation levels when creating traceability chains so that, depending on the selected level, only immediate relations of a requirement or entire traceability chains for the requirement can be generated and displayed. The traceability chains are the foundation for a comprehensive impact analysis during requirements change management.

8.5 Versioning of Requirements

During the life cycle of a system, the requirements of the system change as new requirements are added and existing requirements are removed or altered. The reasons for changes in requirements are diverse. One possible reason is, for instance, the fact that stakeholders learn more and more about the system as requirements engineering progresses. As a result, new

and altered requirements come to their mind. Due to these changes, a suitable versioning of requirements is strongly advisable.

Subject of version control

Versioning of requirements aims at providing access to the specific change states of individual requirements over the course of the life cycle of the system. The version of a requirement is defined by its specific content of the change state and is marked by a unique version number. The information that is subject to version management can be single text-based requirements, sentences, sections of requirements documents, or entire requirements documents, but also requirements models and partial requirements models.

8.5.1 Requirements Versions

When versioning requirements, one can distinguish between the version and the increment of the version number. For example, the version number 1.2 references a requirement with version 1 and the increment 2.

Figure 8-9 illustrates the method of assigning version numbers. As shown in the figure, with smaller changes regarding the content, the increment is increased by one. If larger changes are performed, the version number is incremented. If the version number is increased, the increment is set to the initial value (0). A *v* can be added in front of the version number to make it more understandable and easier to identify as such.

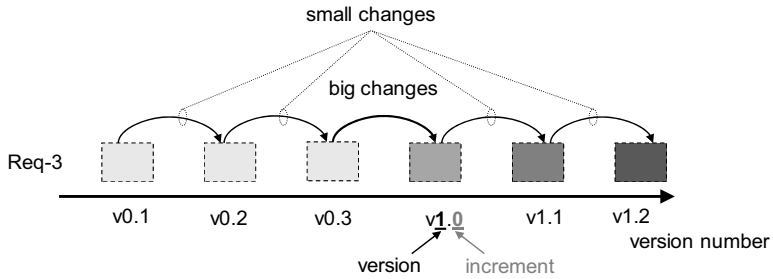

Figure 8-9 *Requirements versions*

Along with the rather simple structuring by means of version numbers, and the proposed method of versioning requirements, other methods of assigning version numbers are widely used. For example, it is possible to distinguish between the version identifier, the increment identifier, and the sub-increment identifier (v1.2.12).

8.5.2 Requirements Configurations

A requirements configuration consists of a set of requirements with the additional condition that each selected requirement is present in the requirements configuration with exactly one version, identified by the version number.

Dimensions of configuration management of requirements

Managing configurations of requirements can be described in two dimensions [Conradi and Westfechtel 1998]: In the product dimension, configuration management deals with individual requirements within the requirements base (foundation). In the version dimension, configuration management considers the various change states as part of version management within the product dimension. Figure 8-10 illustrates both dimensions of configuration management of requirements. On the requirements axis, requirements are represented. On the version axis, the different versions of the requirements are depicted.

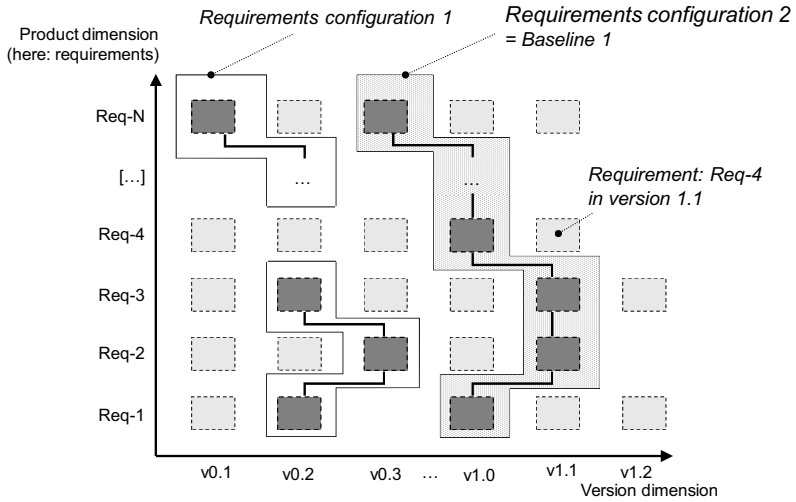

Figure 8-10 *Dimensions of configuration management of requirements (based on [Conradi and Westfechtel 1998])*

Properties of requirements configurations

A configuration of requirements subsumes a defined set of logically connected requirements (more precisely, versions of requirements), where each requirement of the requirements base may occur at most once in the requirements configuration. A requirements configuration does not need to contain a version of every requirement that is considered in the product

dimension (see figure 8-10**,** requirements configuration 1). A configuration of requirements has the following properties:

- *Logical connection:* The requirements contained in a configuration are directly logically connected to one another, i.e., a goal-oriented grouping of the requirements to a common configuration has been performed.
- *Consistency:* The requirements contained in a configuration do not contradict one another, i.e., the configuration contains requirements that are contradiction free in their respective version.
- *Unique identification:* A configuration has a unique identifier (ID) which can be used to uniquely identify the configuration.
- *Immutable:* A configuration defines a certain, immutable state of the specification. If requirements of a configuration are changed, a new version of the requirements and potentially of the configuration is the result.
- *Basis for rollbacks:* If changes of requirements must be undone, configurations offer the ability to roll back requirements to a specific version within a configuration. Therefore, a consistent state of the specification can be maintained.

8.5.3 Requirements Baselines

Configuration vs. baseline

Requirements baselines are specific configurations of requirements that typically comprise stable versions of requirements and, also, often define a release of a system. Due to that property, requirements baselines are usually visible externally (e.g., to the contractor). When requirements baselines are used, a number of important activities in the development process are supported:

- *Basis for release planning:* Requirements baselines are configurations of "stable" requirements, specially marked for the contractor. Baselines therefore serve as the basis of communication for the planning of system releases as well as their definition.
- *Estimation of the effort involved with implementation:* As baselines of requirements can be used for the definition of system releases, they can also be used to estimate the effort needed to realize a system release. This can be done by estimating the partial effort involved with implementing a requirement of the baseline and summing up the total effort for the remaining baseline.

- *Comparison to competing products:* Requirements baselines can be used to compare the planned system to competing systems.

8.6 Management of Requirements Changes

Requirements change over the course of the entire development and life cycle of a system. This means that new requirements are added and existing requirements are changed or removed.

8.6.1 Requirements Changes

Reasons for changes

The reasons for changes in requirements can be multifarious. Along with changes that stem immediately from errors or incomplete requirements, the evolution of the context can make it necessary to change the requirements. For example, changes in the stakeholders' desired application of the system, amendments to a law, new technologies, or additional competition in the market can influence the requirements and make changes necessary. Changes in requirements, however, can also stem from system failure after the system was deployed if an error in the requirements can be held responsible for the failure.

Changes per se are not negative.

Changes in requirements per se are not negative. They are merely an indication that stakeholders deal closely with the system and learn more and more about its functions, qualities, and restrictions. If change requests only occur infrequently during development of the system, it may be a sign of low stakeholder interest in the system to be developed.

Change frequency as an indicator of process quality

However, if requirements changes occur very frequently, the development of a system that is in agreement with all involved stakeholders becomes nearly impossible. A high change frequency is, among other things, an indicator for inadequately performed requirements engineering activities, such as elicitation and negotiation techniques. In addition, a high change frequency takes up a lot of resources in the development project.

8.6.2 The Change Control Board

Over the course of the system life cycle, it is necessary to channel change requests for requirements and define a structured process that will lead to a justified decision about whether a change request is approved and how it

is approved. Changes can pertain to individual requirements (e.g., redefining a requirement) or the entire requirements document. The evaluation of requirements changes, as well as the decision about performing the change, is usually the responsibility of a change control board. The change control board (CCB) typically has the following tasks:

Tasks of the change control board

- Estimate the effort for performing the change (potentially commission a third party with an effort analysis).
- Evaluate change requests, e.g., with respect to the effort/benefit ratio.
- Define requirement changes or define new requirements on the basis of change requests.
- Decide about acceptance or rejection of change requests.
- Classify incoming change requests.
- Prioritize accepted change requests.
- Assign accepted change requests to change projects.

Representatives in the change control board

In some cases, the CCB may want to delegate these tasks to another party. Decisions about changes have to be negotiated and agreed upon with the contractor and all involved stakeholders in the development project. Therefore, the change control board should consist of, among others, the following stakeholders, depending on the properties of the system to be developed and the development process:

- Change manager
- Contractor
- Architect
- Developer
- Configuration manager
- Customer representative
- Product manager
- Project manager
- Quality assurance representative
- Requirements engineer

The role of the change manager

The chairperson of the change control board is the change manager. The change manager has the task, among other things, of mediating between parties in case of conflicts and to negotiate decisions with the respective parties. In addition, the change manager is responsible for communicating and documenting decisions.

8.6.3 The Change Request

Template for change requests

In order to be able to manage changes of requirements during requirements engineering, they have to be documented in a purpose-oriented manner. A change request documents the desired change and contains additional information for the management of the change request.

A change request should contain the following information:

Change information

- *Identifier:* The identifier makes it possible to uniquely identify a change request at any point during the life cycle of the system.
- *Title:* The title summarizes the key concern of the change request in one brief statement.
- *Description:* The description documents the requirement change as precisely as possible. It can contain information on the effect of the changes as well.
- *Justification:* The most important reasons as to why the change is necessary are listed here.
- *Date filed:* The date at which the change request was filed.
- *Applicant:* The name of the person that issued the change request.
- *Priority (in the applicant's opinion):* The importance of the change request according to the applicant's opinion.

Management information for the change request

In addition to the preceding change information, the following information for requirements change management is helpful:

- *Change validator:* The person that verifies if the change has been performed correctly.
- *Impact analysis status:* Flags whether an impact analysis has already been performed on the change request.
- *CCB decision status:* Flags whether the change control board has already decided upon the change request.
- *CCB priority:* Documents the priority of the change request assigned by the change control board.
- *Responsible:* Documents the person that is in charge of performing the change request.
- *System release:* Documents in which system release the changed requirement shall be implemented.

8.6.4 Classification of Incoming Change Requests

Corrective, adaptive, and exceptional changes

After it has been filed, the change request is classified by the change manager and the change control board. Typically, the change manager preclassifies incoming change requests that will be introduced, adapted (if necessary), and finally approved (or rejected) during the next change control board meeting. A change request can be classified according to the following three categories:

- *Corrective requirement change:* A change request is classified thusly if the reason for the change request is a failure of the system during its operation that can be attributed to an error in the requirements.
- *Adaptive requirement change:* A change request is thusly classified if a requested change requires the system to be amended. A possible reason for an adaptive requirement change can be a change in the system context, e.g., a new technology is available or the system boundary was altered (see section 2.2).
- *Exceptional change (hotfix):* A change request is classified as an exceptional change if the change must absolutely immediately be done at all costs. Exceptional changes can be either corrective or adaptive.

Different processing methods

The method for processing requirements changes depends on their classification. For example, exceptional changes must be analyzed, evaluated, decided, and potentially implemented right away. Contrastingly, adaptive requirement changes are often processed in batches at a later point in time, typically as soon as the next (or some subsequent) system release is imminent. On the other hand, corrective requirement changes are usually analyzed, evaluated, and if necessary implemented rather promptly after the change request has been filed.

8.6.5 Basic Method for Corrective and Adaptive Changes

Figure 8-11 illustrates the principal method of handling change requests. This method can be tailored depending on organizational and project-specific particularities.

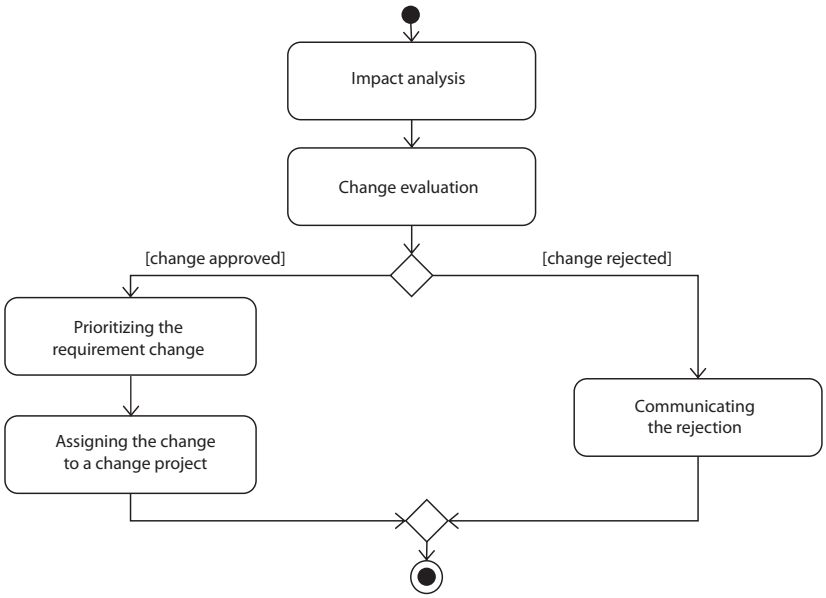

Figure 8-11 *Method for handling change requests*

Impact analysis

During impact analysis, the effort for performing the change is estimated. In order to do so, all requirements affected by the change are sought out, including any newly defined requirements. Afterward, the posterior development artifacts that potentially will have to be changed or redeveloped are identified (e.g., test cases or components). For each affected artifact, the effort for implementing the change is determined and the total effort for the change is computed by summing up all partial efforts.

The consistent integration of the changes into the requirements basis often only negligibly influence the total effort. The most significant portion of the total effort is usually generated by the necessary adaptations of the posterior development artifacts.

Using traceability information

Identifying those requirements and posterior development artifacts that are affected by a requirements change can be automated or at least supported by means of traceability information. If no or not all necessary traceability information is available, domain experts or experts of the development team should be questioned with respect to the consequences of the change request filed.

Evaluating a change

After the impact analysis has been completed, the change control board evaluates the change filed. In order to do that, cost and benefit are compared and evaluated with regard to the available resources. For example, the benefit of the change can be the avoided loss in prestige, improved market position, or avoided contract penalties.

Implementing approved changes

In the next step, approved changes are prioritized by the change control board. Afterward, the requirements changes are assigned to a change project or the next (or any subsequent) system release for implementation.

Validating the requirement changes

Planning, control of the implementation, and validation of the successfully applied changes are typically the responsibility of the change manager or of the change control board and may be delegated, of course.

8.7 Measurement of Requirements

Metrics can be used to assess the quality of requirements and the requirements engineering process. A metric can be used to measure one or more properties of requirements or of the requirements engineering process. The measurement results obtained by using metrics are indicators of the product and process quality.

8.7.1 Product vs. Process Metric

We thus differentiate between two types of metrics:

- Product metrics, used to obtain insights regarding the amount and quality of the documented requirements and requirements documents
- Process metrics, used to obtain insights regarding the progress and quality of the requirements engineering process

8.7.2 Examples of Product and Process Metrics

A typical example of a process metric used in requirements engineering is a metric used to measure the "requirements changes" over a period of time (e.g., already agreed-upon requirements that have been changed within one month or week).

A typical example of a product metric used in requirements engineering is a metric used to measure the "number of requirements errors" identified in a requirements specification at a given point in time. Typically, the error rate is calculated as a relative value, for example, per 100 pages of the specification or per 1000 requirements.

The rate of requirements errors is primarily an indicator of the quality of the requirements documents produced. Moreover, it is also an indicator of the quality of the requirements engineering process.

8.8 Summary

Requirements management is a core activity of requirements engineering. It's the aim of this activity to maintain persistent availability of the documented requirements as well as other relevant information over the course of the entire system or product life cycle, to structure this information in a sensible manner (e.g., by means of requirements attributes), and to ensure selective access to this information. The management of requirements comprises techniques of the following categories:

- *Assigning attributes to requirements:* In order to allow for requirements management, properties of requirements are documented by means of requirements attributes.
- *Prioritizing requirements:* Requirements are prioritized at different points in time, during different activities, and according to different criteria. Depending on the goal of prioritization and the subject of prioritization, different prioritization techniques are to be used.
- *Traceability of requirements:* During requirements management, traceability information of requirements is recorded, organized, and maintained so that information about cross references and dependencies between requirements or between requirements and other development artifacts can be used.
- *Versioning of requirements:* Versioning and configuring requirements makes it possible to keep information about specific developmental states of requirements and requirements documents available over the course of the life cycle of the system or the product.

- *Management of requirements changes:* Usually, the change control board is responsible for processing change requests. The change control board decides if a change request is approved or rejected and prioritizes it. The board also performs an impact analysis to estimate the impact of the change on all requirements and development artifacts as well as the resources necessary for implementing the change.
- *Measurement of requirements*: Product and process metrics can be used to measure the quality of the requirements and the requirements engineering process.