

1 Introduction and Foundations

The impact of requirements engineering (RE) on successful and customer-oriented systems development can no longer be ignored. It has become common practice to provide resources for requirements engineering. In addition, there is a growing understanding that the role of the requirements engineer is essentially self-contained and comprises a series of demanding activities.

1.1 Introduction

According to the figures reported in the Standish Group's Chaos Report of 2006, much has improved in the execution of software projects in the twelve years between 1994 and 2006. While about 30 percent of the software projects investigated in 1994 failed, it was a mere 20 percent in 2006. The number of projects that exceeded time or budget constraints significantly and/or did not meet customer satisfaction dropped from 53 percent to 46 percent [Chaos 2006]. Jim Johnson, chairperson of the Standish Group, names three reasons for the positive development of the figures since 1994. One is that the communication of requirements has much improved since ten years ago. These figures are of importance since how the requirements of a system are handled is a significant cause for project failures and/or time and budget overruns.

Why perform requirements engineering?

1.1.1 Figures and Facts from Ordinary Projects

According to past studies, approximately 60 percent of all errors in system development projects originate during the phase of requirements engineering [Boehm 1981]. These errors, however, are often discovered only in later project phases or once the system has been deployed because incorrect or incomplete requirements can be interpreted by developers in such a fashion that they are subjectively sound or (subconsciously) complete. Missing requirements often remain undetected during design and

Requirements engineering as a cause of errors

realization because developers trust the requirements engineers to deliver high-quality work. Developers implement whatever the requirements document says or what they believe it to be saying. Unclear, incomplete, or wrong requirements inevitably lead to the development of a system that does not possess critical properties or possesses properties that were not requested.

Costs of errors during requirements engineering

The later in the development project a defect in the requirements is corrected, the higher are the costs associated with fixing it. For instance, the effort to fix a requirements defect is up to 20 times higher if the correction is done during programming as opposed to fixing the same defect during requirements engineering. If the defect is fixed during acceptance testing, the effort involved may be up to a 100 times higher [Boehm 1981].

Symptoms and causes of deficient requirements engineering

Symptoms for inadequate requirements engineering are as numerous as their causes. Frequently, requirements are missing or not clearly formulated. For instance, if the requirements do not reflect customer wishes precisely or if the requirements are described in an imprecise way and thus allow for several interpretations, the result is often a system that does not meet the expectations of the client or the users.

The most common reason for deficient requirements is the misconception of the stakeholders that much is self-evident and does not need to be stated explicitly. This results in problems in communication among the involved parties that arise from differences in experience and knowledge. To make matters worse, it is often the case that especially the client wishes for quick integration of recent results into a productive system.

The significance of good requirements engineering

The increasing importance of software-intensive systems in industrial projects as well as the need to bring more innovative, more individual, and more comprehensive systems to market and the need to do so quicker, better, and with a higher level of quality calls for efficient requirements engineering. Complete requirements free from defects are the basis for successful system development. Potential risks have to be identified during requirements engineering and must be reduced as early as possible to allow for successful project progress. Faults and gaps in requirement documents must be discovered early on to avoid tedious change processes.

1.1.2 Requirements Engineering – What Is It?

In order to make a development project succeed, it is necessary to know the requirements for the system and to document them in a suitable manner.

Definition 1-1: Requirement

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

[IEEE Std. 610.12-1990]

The term *stakeholder* is essential in requirements engineering. Among other things, stakeholders are the most important sources of requirements. Not considering a stakeholder often results in fragmentally elicited requirements, i.e., incomplete requirements [Macaulay 1993]. Stakeholders are those people or organizations that have some impact on the requirements. This could be people that are going to interact with the system (e.g., users or administrators), people that have a mere interest in the system but are not likely to use it (e.g., the management, a hacker from which the system must be protected, stakeholders of competing systems), but also legal entities, institutions, etc., because these are embodied by living people who may choose to influence or define the requirements of the system.

Stakeholders

Definition 1-2: Stakeholder

A stakeholder of a system is a person or an organization that has an (direct or indirect) influence on the requirements of the system.

During the development process, requirements engineering must elicit the stakeholders' requirements, document the requirements in a suitable manner, validate and verify the requirements, and manage the requirements over the course of the entire life cycle of the system [Pohl 1996].

Goal of requirements engineering

Definition 1-3: Requirements Engineering

- (1) Requirements engineering is a systematic and disciplined approach to the specification and management of requirements with the following goals:
- (1.1) Knowing the relevant requirements, achieving a consensus among the stakeholders about these requirements, documenting them according to given standards, and managing them systematically
 - (1.2) Understanding and documenting the stakeholders' desires and needs, they specifying and managing requirements to minimize the risk of delivering a system that does not meet the stakeholders' desires and needs

Four core activities of requirements engineering

The four core activities to meet these ends are as follows:

- *Elicitation*: During requirements elicitation, different techniques are used to obtain requirements from stakeholders and other sources and to refine the requirements in greater detail.
- *Documentation*: During documentation, the elicited requirements are described adequately. Different techniques are used to document the requirements by using natural language or conceptual models (see chapters 4, 5, and 6).
- *Validation and negotiation*: In order to guarantee that the predefined quality criteria are met, documented requirements must be validated and negotiated early on (see chapter 7).
- *Management*: Requirements management is orthogonal to all other activities and comprises any measures that are necessary to structure requirements, to prepare them so that they can be used by different roles, to maintain consistency after changes, and to ensure their implementation (see chapter 8).

Constraints

Different project constraints influence requirements engineering. For instance, people, domain factors, or organizational constraints (e.g., spatial distribution or temporal availability of project members) have a large impact on the choice of suitable techniques.

1.1.3 Embedding Requirements Engineering into Process Models

Requirements engineering as a self-contained phase

Ponderous process models (e.g., the Waterfall model [Royce 1987] or the V-Model [V-Modell 2004]) aim at completely eliciting and documenting all requirements in an early project phase before any design or realization decisions are made. The goal of such models is to elicit all requirements

prior to the actual development. As a result, in these process models, requirements engineering is understood to be a finite, time-restricted initial phase of system development.

Lightweight process models (e.g., eXtreme Programming [Beck 1999]), on the other hand, only elicit necessary requirements once they are supposed to be implemented as “foretelling” future functionalities is difficult and requirements change over the course of the project. In these process models, requirements engineering is treated as a continuous, comprehensive process that comprises and integrates all phases of system development.

Requirements engineering as a continuous, collateral process

1.2 Fundamentals of Communication Theory

Requirements must be communicated. In most cases, one uses a rule-driven medium that is accessible to the communication partner—natural language.

Language as a medium for requirement communication

For the transmission of information from one individual to another to work properly, a common code is needed. The sender encodes her message and the receiver has to decode it. Such a common code is intrinsic to any two people that speak the same language (e.g., German), have the same cultural background, and have similar experiences. The more similar the cultural and educational background, the area of expertise, and the everyday work life, the better the exchange of information works. However, such ideal conditions most often do not exist between stakeholders. It is therefore sensible to agree upon a common language and how this common language is to be used. This can, for instance, be achieved by means of glossaries (see chapter 4), in which all important terms are explained. Alternatively, this can be done by agreeing upon a formal descriptive language, e.g., OMG’s Unified Modeling Language, UML (see chapter 6).

Another important factor is the type of communication medium. In verbal communication, the success of the communication relies heavily on redundancy (e.g., language and gestures or language and intonation) and feedback. In written technical communication, for example, information is transmitted with a minimum of redundancy and feedback.

Type of communication medium

In addition to the problems arising from differing domain vocabularies and different communication media, it can often be observed that information is not adequately transmitted or not transmitted at all. This

Language comfort

can be traced back to natural transformations that occur during human perception. These transformational effects are, in particular, *focusing* and *simplification* and can impact the communication more or less harshly.

*Implicit background
knowledge*

Communication—i.e., the language-based expression of knowledge—is necessarily simplifying in nature. The author expects the reader to have some kind of implicit background knowledge. It is the simplifications that arise from language-based knowledge expression that become problematic with regard to requirements, as requirements can become interpretable in different ways. In chapter 5, natural language-based requirement documentation is discussed in further detail.

1.3 Characteristics of a Requirements Engineer

Central role

The requirements engineer as a project role is often at the center of attention. She is usually the only one who has direct contact with the stakeholders and has both the ability and the responsibility to become as familiar as possible with the domain and to understand it as well as possible. She is the one that identifies the needs underlying the stakeholders' statements and amends them in a way that architects and developers—usually laymen where the domain in question is concerned—can understand and implement them. The requirements engineer is, in a manner of speaking, a translator that understands the domain as well as its particular language well enough and also possesses enough IT know-how to be aware of the problems the developers face and to be able to communicate with them on the same level. The requirements engineer therefore has a central role in the project.

*Seven necessary capabilities of
a requirements engineer*

To be able to fulfill all of her tasks, the requirements engineer needs much more than process knowledge. Many of the capabilities required must be based on practical experience.

- *Analytic thinking:* The requirements engineer must be able to become familiar with domains that are unknown to her and must understand and analyze complicated problems and relationships. Since stakeholders often discuss problematic requirements by means of concrete examples and (suboptimal) solutions, the requirements engineer must be able to abstract from the concrete statements of the stakeholder.
- *Empathy:* The requirements engineer has the challenging task of identifying the actual needs of a stakeholder. A core requirement to be able to

achieve this is to have good intuition and empathy for people. In addition, she must identify problems that might arise in a group of stakeholders and act accordingly.

- *Communication skills*: To elicit the requirements from stakeholders and to interpret them correctly and communicate them in a suitable manner, a requirements engineer must have good communication skills. She must be able to listen, ask the right questions at the right time, notice when a statement does not contain the desired information, and make further inquiries when necessary.
- *Conflict resolution skills*: Different opinions of different stakeholders can be the cause of conflicts during requirements engineering. The requirements engineer must identify conflicts, mediate between the parties involved, and apply techniques suitable to resolving the conflict.
- *Moderation skills*: The requirements engineer must be able to mediate between different opinions and lead discussions. This holds true for individual conversations as well as group conversations and workshops.
- *Self-confidence*: Since the requirements engineer is frequently at the center of attention, she occasionally is exposed to criticism as well. As a result, she needs a high level of self-confidence and the ability to defend herself should strong objections to her opinions arise. She should never take criticism personally.
- *Persuasiveness*: Among other things, the requirements engineer is, in a matter of speaking, a kind of attorney for the requirements of the stakeholders. She must be able to represent the requirements in team meetings and presentations. In addition, she must consolidate differing opinions, facilitate a decision in case of a disagreement, and create consensus among the stakeholders.

1.4 Requirement Types

Generally, one can distinguish between three types of requirements:

- *Functional requirements* define the functionality that the system to be developed offers. Usually, these requirements are divided into functional requirements, behavioral requirements, and data requirements (see chapter 4).

Definition 1-4: Functional Requirement

A functional requirement is a requirement concerning a result of behavior that shall be provided by a function of the system.

- *Quality requirements* define desired qualities of the system to be developed and often influence the system architecture more than functional requirements do. Typically, quality requirements are about the performance, availability, dependability, scalability, or portability of a system. Requirements of this type are frequently classified as non-functional requirements.

Definition 1-5: Quality Requirement

A quality requirement is a requirement that pertains to a quality concern that is not covered by functional requirements.

- *Constraints* cannot be influenced by the team members. Requirements of this type can constrain the system itself (e.g., “The system shall be implemented using web services”) or the development process (“The system shall be available on the market no later than the second quarter of 2012”). In contrast to functional and quality requirements, constraints are not implemented, they are adhered to because they merely limit the solution space available during the development process.

Definition 1-6: Constraint

A constraint is a requirement that limits the solution space beyond what is necessary for meeting the given functional requirements and quality requirements.

In addition to the classification into functional requirements, quality requirements, and constraints, a number of different classifications of requirements are used in practice. For example, there are a number of classifications suggested by several standards, e.g., CMMI [SEI 2006] or SPICE [ISO/IEC 15504-5]. Other classification schemes describe requirement attributes, such as the level of detail of a requirement, the priority, or the degree of legal obligation of requirements (see chapters 4 and 8).

1.5 Importance and Categorization of Quality Requirements

In daily practice, quality requirements of a system are often not documented, inadequately documented, or improperly negotiated. Such circumstances can threaten the project's success or the subsequent acceptance of the system under development. Therefore, the requirements engineer should place special emphasis on the elicitation, documentation, and negotiation of quality requirements during the development process.

Typically, many different kinds of desired qualities of the system are assigned to the requirement type *quality requirement*. In order to be able to deal with quality requirements in a structured manner, many different classification schemes for quality requirements have been proposed. The ISO Standard 9126 ([ISO/IEC 9126]), for example, suggests a classification scheme for quality requirements that can also be used as a standard structure for requirements documentation and as a checklist for requirements elicitation and validation. The following list includes some typical categories:

- Requirements that define the quality of system functions, in particular with regard to appropriateness, security and safety, accurateness of calculations, interoperability, and respective conformity to standards
- Requirements that define the dependability of functionalities, in particular with regard to robustness, fault tolerance, and recoverability
- Requirements that define the usability of a system, in particular with regard to understandability, learnability, and ease of use
- Requirements that define the system efficiency, in particular with regard to time behavior (e.g., computation time) or consumption behavior (e.g., resource utilization)
- Requirements that define the changeability of a system, in particular with regard to analyzability, changeability, stability, and testability
- Requirements that define the portability of a system, in particular with regard to adaptability, installability, and respective conformity to standards as well as replaceability

Currently, quality requirements are often specified using natural language. However, numerous approaches to document quality requirements by means of models have been suggested over the past couple of years.

The requirements engineer is responsible for making sure the quality requirements are as objective and verifiable as possible. Typically, this necessitates that the quality requirements are quantified. For example, a quality requirement with regard to system efficiency could specify that a system shall process 95 percent of all queries within 1.5 seconds and that processing queries shall not take longer than 4 seconds at any given time. This can cause quality requirements to be refined by means of additional functional requirements. This could be the case for a quality requirement that is concerned with system security if a functional requirement specifies the exact encryption algorithm to satisfy the need for encryption as demanded by some quality requirement.

Quality requirements are often related to different functional requirements. As a result, quality requirements should always be kept separated from functional requirements. In other words, quality requirements should not be mixed with functional requirements and should be documented separately, with explicit documentation of their relation to functional requirements.

1.6 Summary

Requirements engineering can hardly be avoided, especially when systems are to be developed that satisfy customers and meet budget constraints and schedules. The goal of requirements engineering is to document customer requirements as completely as possible in good quality and to identify and resolve problems in the requirements as early as possible. Successful requirements engineering is based on including the right stakeholders as well as embedding the four core activities of requirements engineering (*elicitation, documentation, validation and negotiation, and management*) into the system development process. At the center of attention is the requirements engineer, who is the primary contact point in requirements engineering and possesses a great deal of domain knowledge and process knowledge as well as a multitude of soft skills.