Requirements Engineering

3

Abstract

This chapter discusses requirements engineering and discusses activities such as requirements gathering, requirements elicitation, requirements analysis, requirements management, and requirements verification and validation.

Keywords

User requirements · System requirements · Functional and non-functional requirements · Requirements elicitation · Requirements analysis · Requirements verification and validation · Requirements management · Requirements traceability

3.1 Introduction

The user requirements specify what the customer wants and define *what* the software system is required to do, as distinct from *how* this is to be done. The requirements are the foundation for the system, and if they are incorrect then irrespective of the best software development processes in the world, the implemented system will be incorrect. The process of determining the requirements, analysing and validating them and managing them throughout the project lifecycle is termed *requirements engineering*.

Often, the initial requirements for a project arise due to a particular problem that the business or customer needs to solve. This leads to a project to implement an appropriate solution, and the first step is to determine the scope of work and the actual requirements for the project, and whether the project is feasible from the cost, time and technical considerations.

G. O'Regan, *Concise Guide to Software Engineering*, Undergraduate Topics in Computer Science, DOI 10.1007/978-3-319-57750-0_3

The *user requirements* are determined from discussions with the customer to determine their actual needs, and they are then refined into the *system requirements*, which state the *functional* and *non-functional* requirements of the system.

The requirements must be precise and unambiguous to ensure that all stakeholders are clear on what is (and what is not) to be delivered, and prototyping may be employed to clarify the requirements and to assist in their definition.

Requirements verification is concerned with ensuring that the requirements are properly implemented (i.e. *building it right*). In other words, it is concerned with ensuring that the requirements are properly addressed in the design and implementation, and a traceability matrix and testing are often employed as part of the verification activities.

Requirements validation (i.e. *building the right system*) is concerned with ensuring that the right requirements are defined and that they are precise, complete, consistent, realizable and reflect the actual needs of the customer. The validation of the requirements is done by the stakeholders, and it involves several reviews of the requirements (and prototype), reviews of the design and user acceptance testing.

The Agile software development methodology (discussed in more detail in Chap. 18) has become very popular in recent years, and its lightweight approach is to be contrasted with the traditional waterfall model. It argues that requirements change so quickly that a requirements document is unnecessary, since such a document would be out of date as soon as it was written.

However, this chapter will focus on requirements engineering as it is in traditional software engineering, and the reader may consult Chap. 18 and the various texts on Agile to understand its approach.

3.2 Requirements Process

The process of determining the requirements for a proposed system involves discussions with the relevant stakeholders to determine their needs and to explicitly define what functionality the system should provide, as well as any hardware and performance constraints.

The specification of the requirements needs to be precise and unambiguous to ensure that all parties involved share a common understanding of the system and fully agree on what is to be developed and tested. A feasibility study may be needed to demonstrate that the requirements are feasible and may be implemented within the defined schedule and cost constraints.

The requirements are the foundation for the system, and project planning is based on the defined requirements. It is therefore essential that the requirements are *complete* (all services required by the user are defined), *consistent* (requirements should not contradict one another) and *unambiguous* (the requirements are clear and definite in meaning). Table 3.1 presents characteristics of good requirements.

Table 3.1 Characteristics of good requirements

No.	Characteristics of good requirements
1.	Each requirement is clear and unambiguous
2.	Each requirement has a priority to indicate its importance
3.	Each requirement may be implemented
4.	Each requirement is testable
5.	Each requirement is necessary
6.	Any conflicts between the requirements are resolved
7.	Each requirement is broken down as fully as possible
8.	Each requirement is consistent with the project's objectives
9.	Each requirement is stated as a stakeholder need (i.e. premature design/solution or implementation information is not included)
10.	The user (business) requirements are traceable (in both directions) throughout the development cycle
11.	The requirements are complete and consistent

Prototyping may be employed to assist in the definition and validation of the requirements, and a suitable prototype will include key parts of the system. It will allow users to give early feedback on the proposed system and on the extent to which it meets their needs. Prototyping is useful in clarifying the requirements and helps to reduce the risk of implementing the incorrect solution.

The implications of the proposed set of requirements need to be understood, as the choice of a particular requirement may affect the choice of another requirement. For example, in the telecommunication domain, two features may work correctly in isolation, but when present together, they interact in an undesirable way. Therefore, feature interactions need to be identified and investigated at the requirements phase to determine how interactions should be resolved.

In situations where an inadequate requirements process is employed, then there may be serious problems in the project. This may be manifested by requirements that are poorly defined or controlled, or requirements that are incomplete, inadequately documented or untestable. In other cases, there may be major scope creep with requirements accepted from any source.

Changes to the requirements may lead to a high level of rework, or cause major delays to the project schedule, or major increases in project cost. In other cases, where poor configuration management practices are employed, the changes to the requirements may not be reflected in the project plan, and the deliverables may be inconsistent with the requirements. Table 3.2 presents symptoms of a poor requirements process:

The following activities are involved in the requirements process, and they are discussed in more detail in the following sections:

- Requirements elicitation and specification
- Requirements analysis
- Requirements verification and validation

Table 3.2 Symptoms of poor requirements process

No.	Symptom
1.	High level of requirements creep during the project
2.	Requirements changing regularly during the project
3.	Missing requirements
4.	Changes to the requirements are not controlled
5.	Requirements accepted from any source
6.	High level of rework during the project
7.	Design, implementation and test products inconsistently interpret the requirements
8.	Deliverables are inconsistent with the requirements
9.	Untestable requirements
10.	Inability to demonstrate that the implementation satisfies the requirements

- Requirements traceability
- Requirements management.

We distinguish between the user (or business) requirements and the system requirements. The *user requirements* are the high-level requirements for the system (they tend to be high-level statements in a natural language with diagrams and tables), whereas the *system requirements* are a more detailed description of what the system is to do. The user requirements are more abstract than the system requirements, and a user requirement is typically expanded into several system requirements. The system requirements provide more detailed information on the system to be implemented, and it details the functionality to be provided and any operational constraints.

The system requirements include the functional and non-functional requirements. A *functional requirement* is a statement about the functionality of the system, i.e. a description of the behaviour of the system and how it should respond to particular inputs. A *non-functional requirement* is a constraint on the functionality of the system (e.g. a timing, performance, reliability, availability, portability, usability, safety, security, dependability or a hardware constraint).

It is essential that the functional and non-functional requirements are stated precisely, and the *non-functional requirements are often quantitatively specified* so that it may be objectively determined (by testing) whether they are satisfied or not. Further, it is essential that the non-functional requirements are satisfied, as otherwise the delivered system may be unusable or unacceptable to the client. The non-functional requirements often affect the overall architecture of the system, rather than the individual components of the system.

Next, we discuss the process of determining the requirements for the system and specifying them in a requirements document.

3.2.1 Requirements Elicitation and Specification

Requirements elicitation is the process of determining the requirements for the proposed system, and it involves discussions with the relevant stakeholders to determine their needs and to explicitly define what functionality the system should provide, as well as any operational and performance constraints. The process of eliciting the requirements from the stakeholders is difficult as

- Stakeholders often do not know what they want from the system.
- Stakeholders often do not know what is or what is not technically feasible and may have unrealistic expectations.
- Stakeholders express the requirements in the language of their domain, which may differ from the language of the business analysts.
- Different stakeholders may want different things from the system resulting in conflicts that need to be resolved.

The project manager/business analyst and the relevant stakeholders will conduct a brainstorming session to define the high-level requirements for the proposed system (or modification to an existing system). The requirements gathering may involve interviews with the stakeholders to allow them to talk about how they currently perform their work and to determine their requirements from the proposed system. It may also include observation session where the business analyst observes the users to see how the work is currently performed.

Further requirements workshops will review and analyse the draft user and system requirements documents and identify all other relevant information for the proposed system. There will typically be two requirements documents produced, and these are the *user* (sometimes called business) *requirements specification* (URS or BRS) and the *system requirements specification* (SRS). These two documents could potentially be combined into one document (Fig. 3.1).

The user requirements document is usually written in a natural language such as English (it may include diagrams and tables), and it describes the external behaviour of the system and specifies the functional and non-functional requirements in non-technical language. The systems requirements document will be an expanded version of the user requirements, and it provides the detail as to how the user requirements are provided in the system. It is a detailed specification of the entire system, with the aim of describing the external behaviour of the system and excluding (as far as possible) design information.[1] The SRS may be written in:

[1]It is desirable that the user or system requirements describe what is to be provided rather than how it is to be provided. That is, in theory, design or implementation information should be excluded in the specification. However, in practice, it is sometimes difficult to exclude all design information (e.g. consider the case where a system needs to work with an existing system).

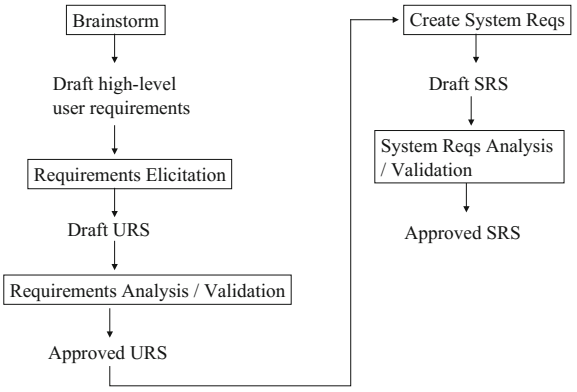

Fig. 3.1 Requirements process

- A natural language
- A graphical language
- Formal specification language.

The system specification is generally written in a natural language such as "*English*" (with diagrams and tables included). Natural language is inherently ambiguous, and therefore, care is required to ensure that the definition is precise and unambiguous, and the specification needs to be carefully reviewed to ensure that any ambiguities are identified and removed.

The specification may be written in a graphical specification language such as UML, which is often employed in defining the functional requirements of a system using use case diagrams, state diagrams and sequence diagrams. Finally, extra precision is needed for the specification of the requirements in the safety critical and security critical fields, and a formal mathematical specification language (such as VDM or Z) is often used in these domains.

Prototyping may be employed, and it helps in identifying gaps and misunderstandings in the definition of the requirements. The prototype is an early working version of the system, it is used to give the users a flavour of what the working system will look like, and its evaluation by the stakeholders helps in clarifying the requirements. The prototype may be thrown away at the end of prototyping, or it may be reused in the development of the system. Prototyping involves:

- Define prototype objectives
- Decide which functional requirements will be prototyped
- Develop the prototype
- Evaluate the prototype.

The project manager (or a business analyst) will facilitate the requirements workshops, and the initial workshop is an interview and brainstorming session[2]

[2]It may involve getting end-users to talk about how they currently do a certain task and brainstorming on better ways in which the proposed system can do the same task.

focused on *requirements discovery*. This involves identifying and gathering the requirements from the various stakeholders, analysing and prioritising them, resolving conflicts between them and consolidating them into a coherent set of user requirements.

This leads to the first draft of the user requirements, which is prepared by the project manager/business analyst, and the draft document is circulated to the stakeholders for review and comments. Further requirements workshops are then held to discuss and analyse the current draft of the user requirements, to ensure that they meet the needs of the stakeholders, as well as identifying new requirements and resolving any conflicts.[3] This process continues until all stakeholders are in agreement with the user requirements and are prepared to approve them. In some cases, the user requirements may already be defined and documented by the customer.

The project manager/business analyst may employ a checklist as an aid to determine that the requirements process has been followed and to verify that the user requirements have been fully specified and that every requirement specified is actually necessary. The final version of the user requirements document is circulated to all participants for their final review and approval.

Once the user requirements have been approved by all stakeholders, the work on the *system requirements* commences, and the business analyst expands the user requirements into more specific and detailed system requirements. Several workshops/reviews of the system requirement specification take place with the stakeholders, with the goal of ensuring that the system requirements are valid with respect to the business requirements and that they meet stakeholders' needs and are fit for purpose. Finally, the stakeholders approve the SRS.

Scenarios are useful in adding detail to the requirements, with each scenario covering a small number of possible interactions with the system. Use cases are often used to identify the actors involved in the interactions, and they provide a useful way to elicit the requirements from the stakeholders who interact directly with the system.

The ambiguity of natural language has led to interest in more precise notations to express requirements unambiguously. We mentioned the graphical unified modelling language (UML) [1], which has become popular in recent years. Its use case diagram is often used for requirements elicitation, with the use cases (Fig. 14.2) describing the functional requirements of the system graphically. The use cases describe the scenarios (or sequences of actions) in the system from the user's viewpoint (actor). It shows how the actor interacts with the system, where an actor represents the set of roles that a user can play, and the actor may be human or an automated system. Use case diagrams and various UML diagrams are discussed in Chap. 14.

[3]Conflicts are inevitable as stakeholders will have different needs, and so discussion and negotiation are required to resolve these conflicts and achieve consensus.

Formal specification notations such as Z or VDM are often employed in the safety critical or security critical fields. The advantage of these mathematical languages is that they are precise and amenable to proof, and mathematical analysis may be employed in a sense to debug[4] the requirements. This provides increased confidence in the correctness and validity of the requirements. However, these notations are perceived as being difficult to use by industrialists, and they are not widely employed in mainstream software engineering. Formal methods are discussed in more detail in Chap. 12.

3.2.2 Requirements Analysis

The requirements analysis activities are conducted as part of requirements elicitation, and the requirements are analysed to ensure that they are those that are actually required; that they are precisely and unambiguously stated; that they are complete and consistent; that they are categorized and prioritised; and that any conflicts between them are identified and resolved. There may be an initial feasibility study prior to the commencement of the project to ensure that the proposed system is technically feasible and achievable within the defined budget and time constraints.

The resolution of any conflicts is through discussion and negotiations with the stakeholders. The requirements are generally prioritised to define the importance of each requirement, and a number of development models (e.g. the Rational Unified Process) implement the most important requirements first. Requirements analysis is an iterative process with feedback going back to the stakeholders in the requirements elicitation process.

The requirements workshops will verify that the system requirements are valid with respect to the user requirements, and technical workshops will need to be conducted to determine the appropriate approach to their implementation.

3.2.3 Requirements Verification and Validation

The difference between requirements validation and verification is illustrated by the phrase "*Building the right thing*" versus "*building it right*". In other words, *validation* is concerned with ensuring that the correct requirements are being implemented, whereas *verification* is concerned with ensuring that the defined requirements are being implemented correctly.

The stakeholders *validate* the requirements to ensure that they are the right set of requirements and that their implementation will result in a system that is fit for purpose. It is essential to validate the requirements, as the cost of correction of a requirements defect increases the later that the defect is discovered. Therefore, it is essential to identify a requirements defect as early as possible, as otherwise there

[4]Essentially, the mathematical language provides the facility to prove that certain properties are true of the specification, and that certain undesirable properties are false in the specification.

may be major cost and time involved in its correction, especially if the defect is discovered late in the software development lifecycle.

The validation activities may involve checks that the requirements are complete, consistent, feasible, testable and are fit for purpose. The validation may involve prototyping and several reviews (and updates) of the requirements (and prototype) by the stakeholders, until all stakeholders are ready to approve the requirements of the system.

The validation of the requirements will ensure that the requirements are complete and consistent, as well as reflecting the needs of the customer. The final validation step is the user acceptance testing, and this is performed by the customer to confirm that the completed system is fit for purpose and satisfies customer expectations. The lifecycle model employed determines the verification and validation activities to be conducted during the project, with models such as joint application development (JAD) and Agile involving a high level of customer involvement throughout the lifecycle.

Requirements verification is concerned with ensuring that the system as built (from design, to implementation, to testing and deployment) properly implements the defined requirements. A traceability matrix (Table 3.4) shows how the requirements are implemented and tested, and it may be employed as part of requirements verification.

It shows how the user requirements have been addressed in the system requirements, and how they have been implemented in the design of the system, as well as showing how the test cases have verified that the implementation has implemented the requirements correctly.

3.2.4 Requirements Managements

Requirements management is concerned with managing changes to the requirements, and in ensuring that the project maintains an up-to-date approved set of requirements throughout the project lifecycle. It is essential that the project deliverables are kept consistent with the latest version of the requirements, and that when the requirements document changes then all other project deliverables such as the design document, software modules and test specifications are kept consistent with the new version of the requirements.

It is an important area to get right as all project activities are planned from the approved set of requirements. Requirements management is concerned with managing changes to the requirements of the project, and in identifying inconsistencies between the requirements and the project plans and work products. Its focus is on the *activities for managing changes to the requirements*, as distinct from the activities in gathering and defining the requirements.

It is important that changes to the requirements are controlled, and that the impacts of the changes are fully understood prior to authorization. Once the system requirements have been approved, any proposed changes to the requirements are subject to formal change control. The project will set up a group that is responsible

for authorizing changes to the requirements [usually called the *change control board* (CCB)]. The CCB is responsible for analysing requests to change the requirements, and it makes an informed decision on whether to accept or reject the change request based on its impacts and risks.

The need to change the requirements may be due to business or regulatory changes, or to a customer need becoming apparent at a late stage of the project when the project is nearing completion. A request to change the requirements is termed a *change request* (CR), and this is a stakeholder request for a change to the scope of the project, and it may arise at any time during the project. The impacts of the CR (e.g. technical, risks, cost, budget, and schedule) need to be carefully considered, as a change introduces new risks to the project, and may adversely affect cost, schedule and quality.

Therefore, it is essential that the impacts of the CR be fully considered prior to its authorization. The CR is considered by the CCB, and an informed decision is made to authorize or reject the request. The activities involved in managing change requests are summarized in Table 3.3.

Following the approval of a CR, the affected documents such as the system requirements, the design and software modules are modified accordingly. This is done to ensure that all of the project deliverables are kept consistent with the latest version of the requirements. Testing is carried out to verify that the changes have been implemented correctly.

3.2.5 Requirements Traceability

The objective of requirements traceability is to verify that all of the defined requirements for the project have been implemented and tested. One way to do this is to consider each requirement number and to go through every part of the design document to find where the requirement is being implemented in the design and similarly to go through the test documents and find any reference to the requirement number to show where it is being tested. This would demonstrate that the particular requirement number has been implemented and tested.

A more effective mechanism to do this is to employ a traceability matrix, which may be employed to map the user requirements to the system requirements; the system requirements to the design; the design to the unit test cases; the system test cases; and the UAT test cases. That is, traceability is defined through the project lifecycle, and the matrix provides a crisp summary of how the requirements have been implemented and tested.

The traceability of the requirements is *bidirectional*, and the traceability matrix may be maintained as a separate document, or as part of the requirements document. The basic idea is that a mapping between the requirement numbers and sections of the design or test plan is defined, and this provides confidence that all of the requirements have been implemented and tested.

Requirements will usually be numbered, and a single requirement number may map on to several sections of the design or to several test cases, i.e. the mapping is

Table 3.3 Managing change requests

Activity	Change request
Log change request	The change request is logged and a unique reference number and priority assigned
Assess impact	The cost, schedule, technical and quality impacts are determined and the risks identified
Decision	The CCB authorizes or rejects the change request
Implement solution	The affected project documents and software modules are identified and modified accordingly
Verify solution	Testing (unit, system and UAT) is employed to verify the correctness of the solution
Close CR	The change request is closed

often *one to many*. The traceability matrix (Table 3.4) provides the mapping between individual requirement numbers, and the sections in the design or test plan corresponding to the particular requirement number.

It is essential to keep the traceability matrix up to date during the project and especially after changes to the requirements. The traceability matrix is useful as a tool whenever there are changes to the requirements as it allows the impacts of the change on the other requirements (and other project deliverables) to be easily determined.

3.3 System Modelling

A model is an abstraction (simplification) of the physical world, and it acts as a representation of reality. The aim of the model is to capture the essential details of the real world, and as it is a simplification of the reality, it does not include all aspects of the physical world. However, it is important that all of the key aspects to be studied are included in the model and to determine the adequacy of the model as a representation of the real world.

A model is considered suitable if its properties closely match those of the system being modelled. It is common to employ models in engineering: for example, in civil engineering, it is normal to develop models of bridges and traffic flow prior to constructing a bridge. These models help in understanding the anticipated stresses on the bridge and play an important role in the design of a bridge that is safe to use. It is important that the models are an adequate representation of the reality, as otherwise there is the potential for serious consequences. For example, the model of the Tacoma Narrows Bridge did not include aerodynamic forces, and this proved to be a major factor in its subsequent collapse [2].

A good model will allow predictions of future behaviour to be made, and the adequacy of the model is determined from model exploration. This involves asking questions and determining the extent to which the model provides accurate answers

Table 3.4 Sample trace matrix

Requirement no.	Sections in design	Test cases in test plan
Rl.l	D1.4, D1.5, D3.2	T1.2, T1.7
R1.2	D1.8, D8.3	T1.4
R1.3	D2.2	T1.3
R1.50	D20.1, D30.4	T20.1, T24.2

to the questions. Inadequate models are replaced over time with better models that provide a better explanation of the reality. For example, the Ptolemaic cosmological model was replaced by the Copernican model, and Newtonian mechanics was replaced the theory of relativity when dealing with velocities that are close to the speed of light [3].

The adequacy of the model will determine its acceptability as a representation of the physical world. Models that are ineffective will be replaced with models that offer a better explanation of the manifested physical behaviour.

Occam's Razor[5] ('*principle of parsimony*') is a key principle employed in modelling [4]. It states that the number of entities employed to explain the reality should be kept to a minimum, with every entity used actually required for the explanation. In other words, the simplest model should be chosen with the least number of assumptions, and all superfluous concepts that are not required to explain the phenomena should be removed. This results in a crisp and simpler model.

System modelling is an abstraction of the existing and proposed system, and it helps in clarifying what the existing system does and in communicating and clarifying requirements of the proposed system. The model is a simplification of the system, and it may be explored to identify strengths and weaknesses in the existing system. This leads to requirements for the new system.

Models of the new system may be used to communicate the proposed requirements to the other stakeholders, and more than one model (e.g. using several UML diagrams) may be employed to represent the system from a number of different viewpoints (e.g. environment, behaviour, structural or behaviour). The use of the graphical UML diagrams to represent the software system is a useful type of system modelling.

Another important approach (used mainly in the safety and security critical field) is the use of mathematical models that provide abstract mathematical models of the proposed software system.

Model-driven engineering is concerned with the generation of the programs from the models, and the Rational/IBM tools allow programs to be generated from the UML diagrams.

[5]This principle is named after the medieval philosopher, William of Ockham.

3.4 Review Questions

1. What is the difference between a functional and non-functional requirement?
2. What is the difference between requirements verification and validation?
3. What is requirements engineering? How are requirements elicited from the customer?
4. Explain the difference between a user requirement and a system requirement?
5. How are changes to the requirements managed? Why is it important to keep project deliverables consistent with the requirements?
6. What is the purpose of requirements traceability?
7. Explain the advantages and disadvantages of specifying the system requirements in a natural language. Describe other approaches.
8. Explain the purpose of a model and how models may be used in requirements engineering.

3.5 Summary

The user requirements specify what the customer wants and define what the software system is required to do, as distinct from how this is to be done. The requirements are the foundation for the system, and so if they are incorrect, then the implemented system will be incorrect. The process of determining the requirements, analysing and validating them and managing them throughout the project lifecycle is termed requirements engineering.

The user requirements are determined from discussions with the customer to determine their actual needs, and they are then refined into the system requirements, which state the functional and non-functional requirements of the system. The requirements must be precise and unambiguous to ensure that all stakeholders are clear on what is (and what is not) to be delivered.

Prototyping may be employed to assist in the definition of the requirements. Requirements verification is concerned with ensuring that the requirements are properly implemented, and it is concerned with ensuring that the requirements are properly addressed in the design and implementation. A traceability matrix and testing are often employed as part of the verification activities.

Requirements validation is concerned with ensuring that the right requirements are defined and that they are complete, consistent and reflect the actual needs of the customer. The validation of the requirements is done by the stakeholders, and it

involves several reviews of the requirements (and prototype), reviews of the design and user acceptance testing.

Requirements management is concerned with managing changes to the requirements, and in ensuring that the project maintains an up-to-date approved set of requirements throughout the project lifecycle. It ensures that the project deliverables are kept consistent with the latest version of the requirements, and when the requirements document changes, then all other project deliverables need to be kept consistent with the new version of the requirements.

The objective of requirements traceability is to verify that all of the defined requirements for the project have been implemented and tested. The traceability matrix provides a crisp summary of how the requirements have been implemented and tested, and it provides a bidirectional mapping of the requirements to the design and test case.

References

1. I. Jacobson, G. Booch, J. Rumbaugh, *The Unified Software Modelling Language User Guide* (Addison-Wesley, Reading, 1999)
2. G. O'Regan, *A Practical Approach to Software Quality* (Springer, New York, 2002)
3. T. Kuhn, *The Structure of Scientific Revolutions* (University of Chicago Press, Chicago, 1970)
4. M.M.A. Airchinnigh, Conceptual models and computing. PhD Thesis. Department of Computer Science, University of Dublin. Trinity College, Dublin, 1990