Software Engineering

Lecture 5

Requirements Engineering

Dr. Obuhuma James

Description

This topic focuses on the process of requirements engineering. It outlines what requirements are and the various categories of requirements. The entire process of gathering requirements has also been outlined. Finally, the structure and components of a requirements specification document will also be covered as a main deliverable of the requirements engineering phase.

Learning Outcomes

By the end of this topic, you will be able to:

- Differentiate between the various types of requirements.
- Describe the requirements engineering process.
- Develop a requirements specification document for any systems development case.

Overview of Requirements Engineering

Requirements engineering refers to the process of determining the services that users need out of a systems. This also includes the constraints under which the system is to be developed and used. The previous topics introduced the requirements engineering process where it was determined to be encompassing four development activities, namely, feasibility study, requirement elicitation and engineering, requirement specification, requirement validation. It may however be accomplished differently depending on whether you are using the plan-driven methodology or the agile methodology. For instance, due to the detailed nature of the plan-driven approach, during the requirements engineering phase, an agreed requirements document that specifies a system satisfying stakeholder requirements is generated. At this point, requirements are usually presented at two levels of detail, namely, end-user needs, and system developer needs. Generally, end-users and customers need a high-level statement of the requirements while system developers need a more detailed system specification. On the other hand, in the case of agile development, the requirements engineering phase is fully interleaved with the design and implementation phase. This means that the requirements engineering phase does not have to result into a detailed system specification document. The user requirements document hence only define the most important characteristics of the system.

Types of Requirements

Requirements can be visualized as descriptions of the system services and the constraints under which the system will be developed and operate. Requirements are normally generated through a process referred to as requirements engineering. Such descriptions may range from a high-level abstract statement of services to be accomplished by the system or constraints all the way to mathematical functional specifications. In a nutshell, the level of details of requirements depends on different factors including the purpose to be served by the requirements.

According to Sommerville [1], there are two broad types of requirements

1. User requirements

This are high level abstract statements and diagrams of the services the system should provide and its operational constraints. They are normally expressed in natural language. User requirements are mainly written for the customer (the end user). However, part of the large target group includes client managers, client engineers, contract managers, and system architects. User requirements are often referred to as user needs since they describe the activities the user should be able to perform in system. User requirements are generally approved by the user and used as the basis for the determination of the desired system requirements.

2. System requirements

This are the main building blocks that guide developers as they build the system. Thus, they define what should be implemented by the developer. In most cases, they are normally part of the binding contract between clients and contractors. They are normally written for the software developers and others including system end users, client engineers, and system architects. System requirements are normally broadly categorized as either functional or non-functional requirements.

It is sometimes quite hard for users to speak out what exactly they need the system to do for them. Thus, the need for having a business analyst who can aid in thorough analysis of user requirements followed by careful formulation and documentation of well-focused system requirements.

Functional and Non-functional Requirements

Functional requirements are statements of the services that the system should provide, how the system should react to particular inputs and how the system should behave in particular situations. Functional requirements focus on bringing out the basic system behaviour by showing what the system does. They can be visualized in terms of how the system responds to given inputs. They may also state what the system should not do. Thus, in case a system fails to meet its functional requirements, then it will be deemed as not working.

On the other hand, non-functional requirements lay out the constraints on the services or functions offered by the system. This includes timing constraints, constraints of the development process, standards among others. Non-functional requirements specify how the system should do its functional requirements, without affecting the basic functionality of the system. Even if the non-functional requirements are not met, the system will still perform its basic purpose but will fail on usability which is a critical non-functional requirement. Non-functional requirements define system behaviour, features, and general characteristics that affect the user experience. The ease of system use relies on how well non-functional requirements are defined and executed. Hence, non-functional requirements offer a mechanism for judging system performance.

A major deference between functional and non-functional requirements is that functional requirements bring out features of the product and are focused on user requirements while non-functional requirements bring out the properties of the product and are focused on user expectations. For instance, if we have a functional requirement like “The system must generate daily sales report upon clicking on the daily reports button”, there must be a corresponding non-functional requirement that specifies how fast this should happen. Despite the fact that the functional requirement will be met, missing such a non-functional requirement renders user experience and perception of quality at risk for they may have to wait for unexpectedly longer durations.

Non-functional requirements can be expressed in three classes, namely, product requirements, organizational requirements, and external requirements as shown in Figure 1.

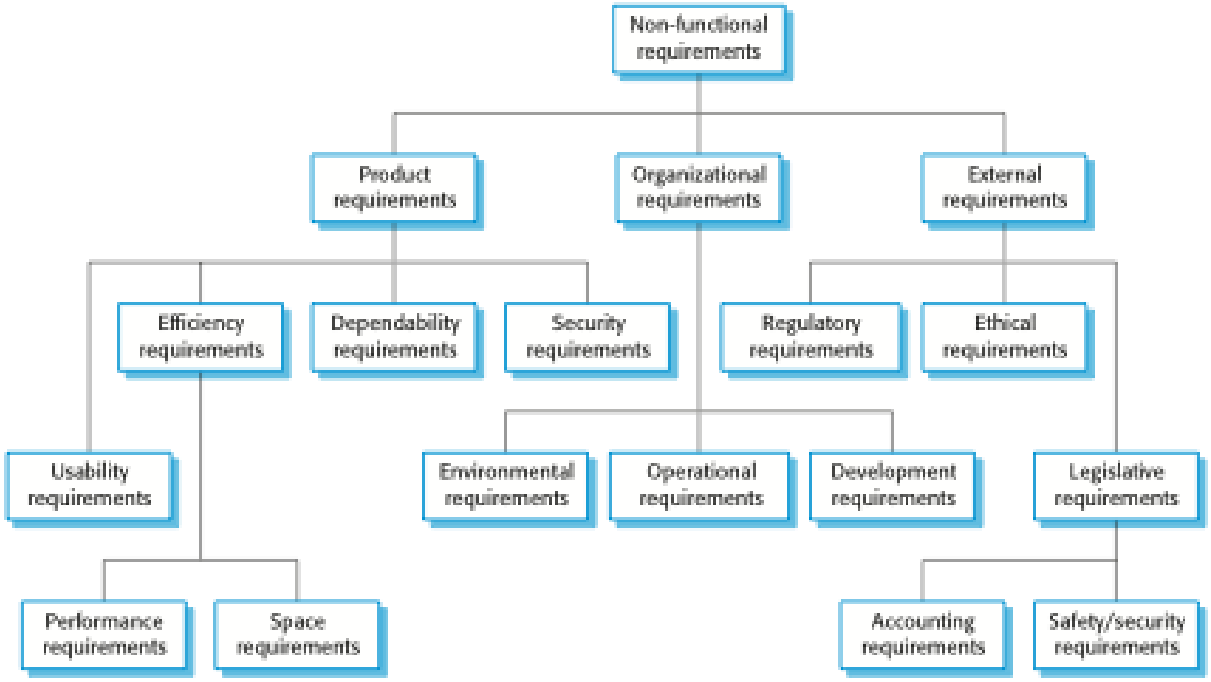

Figure 1. Classification of Non-Functional Requirements [1]

Based on Figure 1, product requirements specify a particular behaviour that the product must take leading to usability, efficiency, dependability, and security. Organization requirements result from organizational policies and procedures that could be viewed in terms of environmental, operational and development requirements. Finally, external requirements arise from factors that are external to the systems and its development process. These include regulatory, ethical, and legislative requirements. In most cases, non-functional requirements are normally aligned to the following metrics for ease of measurement [1]:

1. Speed – includes a measure of processed transactions per second, response time, refresh time, among others.
2. Size – includes a measure of amount of memory, among others.
3. Ease of use – includes a measure of training time, number and type of support requests, among others.
4. Reliability – includes a measure of mean time to failure, probability of unavailability, rate of failure occurrence, availability, among others.
5. Robustness – includes a measure of restart time from failure, percentage of events leading to failure, probability of data corruption on failure, among others.
6. Portability – includes percentage of target dependent statements, number of target systems, among others.

A third and slightly silent type of system requirements are the domain requirements which are the constraints on the system with respect to the domain (or area) of operation. The system may end up being unworkable in cases where domain requirements are not satisfied.

Table 1 and 2 shows examples of how user requirements, system requirements, functional requirements and non-functional requirements can be stated.

Table 1. Examples of User Requirements with Corresponding System Requirements

User Requirements	Systems Requirements
1. The system shall generate daily reports showing student class attendance	1.1 At 5:00pm each day, a report will automatically be emailed to the unit lecturer showing the attendance summary 1.2 Changes to attendance records shall be restricted to the unit lecturers 1.3 The registrar staff will have access to all class attendance reports for any given day

Table 2. Examples of Functional and Non-Functional Requirements

Functional Requirements	**Non-Functional Requirements**
The system shall allow users to login upon provision of valid login credentials	The system must process each user request within 10 seconds
All changes made on the data shall be logged by the server	The system shall be able to accommodate 2000 concurrent user at a time
The system shall only allow administrative staff to view students' grades upon login	The system shall always maintain the privacy of user data

Please note that it is best practice to consider having non-functional requirements cut across the various classifications.

Software Requirements Document

Requirements are formally spelt out in a software requirements document. The document should be made in such a manner that shows WHAT the system should do rather than HOW it should do it. As earlier mentioned, the need and level of details of the information in such documents depends on the type of system and development methodology used. For instance, such a document may not be helpful in situations where agile methodologies are used as it may be deemed a waste of time due to the rapidly changing requirements. According to Sommerville [1], the requirements documents should contain the following sections:

1. Preface
2. Introduction
3. Glossary
4. User requirements definition
5. System Architecture
6. System requirements specification
7. System models
8. System evolution
9. Appendices
10. Index

The main consumers of the requirements document are system customers, managers, system engineers, system test engineers, system maintenance engineers, among others.

Requirements Specification

Requirements specification refers to the process of writing done the user and system requirements in a requirements document. For better system specification, requirements should be both complete and consistent. Under all circumstances, user requirements should be easy to comprehend by end users who lack technical knowledge while system requirements should be more detailed and may include more technical aspects that can be consumed by system developers. Specification of system requirements may include the use of natural language, structured natural language, design description languages, graphical notations, mathematical specifications, among others.

Requirements Engineering Process

The process involved in requirements engineering may vary depending on the application domain, the people involved and the organization working on the requirements. However, the main tasks undertaken involves elements of feasibility study, requirements elicitation and analysis, requirements specification, requirements validation, and requirements management as summarized in Figure 2. Above all, there should some way to take care of requirements management.

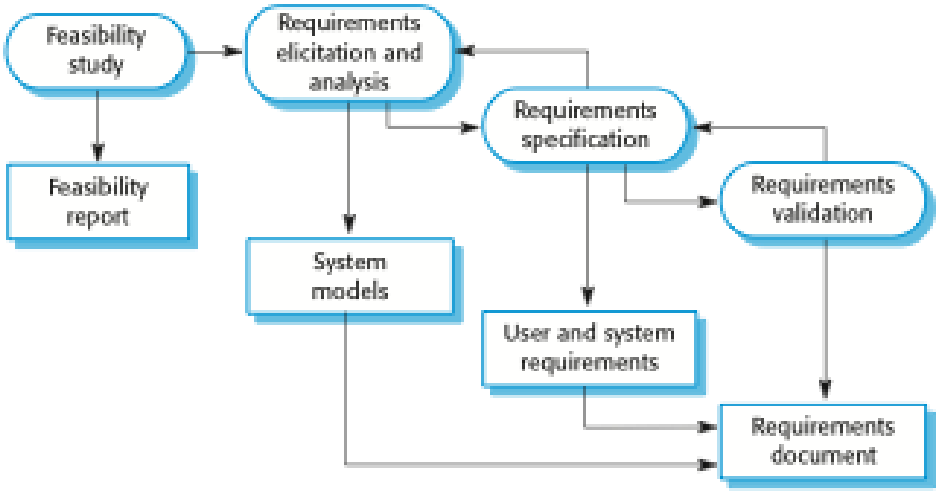

Figure 2. Requirements Engineering Process [1]

Due to the iterative nature of the requirements engineering process, the various process activities are normally interleaved. The following is a brief description of each of the activities:

1. Feasibility Study

The feasibility study phase establishes a proper reason for developing software that is acceptable to users, flexible to change and conformable to established standards. The are three types of feasibility, namely,

a) Technical feasibility – an evaluation of the existing technologies required to accomplish customer requirements for the given time and budget.
b) Operational feasibility – an evaluation of the range in which the anticipated software performs its task based on customer requirements.
c) Economic feasibility – an evaluation of whether the anticipated software can generate financial profits for an organization. This also includes an analysis of the cost of implementing the given solution.

2. Requirements Elicitation and Analysis

This is the phase when the requirements are gathered from the customers and/or existing systems. Requirements analysis immediately starts as the elicitation is ongoing. Accuracy in the process depends on a number of factors, including,

a) Involvement of the right people
b) Having a business analyst to clearly bring out what stakeholders express
c) How requirement changes are accommodated at the analysis stage

d) Mechanisms to manage organizational and political factors

3. Requirements Specification

This phase is as described in the previous section which in summary entails the writing done of the requirements in a requirements document.

4. Requirements Validation

The phase entails checking and verifying that the requirements as stated in the document are clear and aligned to user needs. For instance, the following are some guiding questions for requirements validation:

a) Are the requirements practically implementable?
b) Are they geared towards meeting user requirements?
c) Are they clear with no ambiguities?
d) among others.

Some of the techniques used in requirements validation include reviews and inspections, prototyping, use of test cases, automated consistency analysis, among others.

5. Requirements Management

This phase is not part of the activities shown in Figure 2. However, it is a critical process that aids in the management of changing requirements during the requirements engineering process and actual systems development. It is worth noting that change is inevitable. Requirements management however depends entirely on the software development methodology being used. This is as covered in topic 3 and 4 on plan-driven and agile software development respectively. Reference should be made to what was covered in the two topics.

Summary

The topic has extensively covered the entire scope of requirements that are quite critical in subsequent phases of the system development process. The two broad categories of requirements, namely, user and system requirements, have been covered at length. Furthermore, non-functional requirements have further been explored under three classes, namely, product, external and organizational requirements. The element of usability has been covered under the product class of requirements. The topic has also outlined the basic

components of a requirements document that act as a contract between the customer and the developer. The requirements document acts as a the end product of the requirement engineering phase, whose entire process has also been covered.

Check Points

1. Differentiate between the two broad types of requirements.
2. Differentiate between functional and non-functional requirements.
3. Describe the three classifications of non-functional requirements.
4. Discuss the process of requirements engineering.
5. Distinguish between requirements engineering and requirements specification.
6. Describe the components of a requirements document.

Learning Resources

Core Textbooks

1. Sommerville, I., Software Engineering, 10th Edition, Addison Wesley, 2016.

Other Resources

2. Pries, K. H., & Quigley, J. M. (2010). Scrum project management. CRC press.
3. Schwaber, K. (2004). Agile project management with Scrum. Microsoft press.

References

[1] Sommerville, I., Software Engineering, 10th Edition, Addison Wesley, 2016.

[2] Pries, K. H., & Quigley, J. M. (2010). Scrum project management. CRC press.

[3] Schwaber, K. (2004). Agile project management with Scrum. Microsoft press.

[4] Gotterbarn, D., Miller, K., & Rogerson, S. (1997). Software engineering code of ethics. Communications of the ACM, 40(11), 110-118.

[5] Pressman, R. S. (2005). Software engineering: a practitioner's approach. Palgrave macmillan.

[6] Kendall, K. E. and Kendall, J. E., Systems Analysis and Design, 8th Edition, Pearson, 2011.