*A requirement is a feature of the system or a description of something the system is*

*capable of doing in order to fulfill the system’s purpose.*

Figure 3.1 illustrates the process of determining the requirements for a

software-based system.

Requirements describe the “what” of a system, not the “how.” Requirements

engineering produces one large document, written in a natural language, and

contains a description of what the system will do without describing how it will

do it.

Requirements engineering is the systematic use of proven principles,

techniques, and language tools for the cost-effective analysis, documentation,

and on-going evaluation of the user’s needs and the specifications of the external

behavior of a system to satisfy those user needs. It can be defined as a discipline,

which addresses requirements of objects all along a system-development process.

**53**

**54** SOFTWARE ENGINEERING AND TESTING

**FIGURE 3.1** The Process of Determining Requirements

The input to requirements engineering is the problem statement prepared by

the customer. The output of the Requirements Engineering (RE) process is a system

requirements specification called the Requirement Definition and Description

(RDD). The system requirements specification forms the basis for designing

software solutions.

**3.1.1 Types of Requirements**

There are various categories of the requirements. On the basis of their priority, the

requirements are classified into the following three types:

\_ Those that should be absolutely met.

\_ Those that are highly desirable but not necessary.

\_ Those that are possible but could be eliminated.

On the basis of their functionality, the requirements are classified into the

following two types:

(*i*) ***Functional requirements.*** They define factors, such as I/O formats, storage

structure, computational capabilities, timing, and synchronization.

(*ii*) ***Non-functional requirements.*** They define the properties or qualities of

a product including usability, efficiency, performance, space, reliability,

portability, etc.

INTRODUCTION TO SOFTWARE REQUIREMENTS SPECIFICATION **55**

**3.2 PROCESS OF REQUIREMENTS ENGINEERING**

Figure 3.2 illustrates the process steps of requirements engineering.

**FIGURE 3.2** Process Steps of Requirements Engineering

Requirements engineering consists of the following processes:

\_ Requirements gathering (elicitation).

\_ Requirements analysis and modeling.

\_ Requirements documentation.

\_ Requirements review.

\_ Requirements management.

**3.2.1 Requirement Elicitation and Analysis**

**Requirement Elicitation/Gathering.** Requirement gathering is a communication

process between the parties involved and affected in the problem situation. The

tools in elicitation are meetings, interviews, video conferencing, e-mails, and

existing documents study and facts findings. More than 90% to 95% elicitation

should be complete in the initiation stage while the remaining 5% is completed

during the development life-cycle. The requirements are gathered from various

sources. The sources are:

\_ Customer (Initiator)

\_ End Users

\_ Primary Users

\_ Secondary Users

\_ Stakeholders

**Requirement Analysis.** Requirement analysis is a very important and essential

activity after elicitation. In this phase, each requirement is analyzed from the

point-of-view of validity, consistency, and feasibility for firm consideration in the

RDD and then in the SRS. Validity confirms its relevance to goals and objectives

and consistently confirms that it does not conflict with other requirements but

supports others where necessary. Feasibility ensures that the necessary inputs are

**56** SOFTWARE ENGINEERING AND TESTING

available without bias and error, and technology support is possible to execute the

requirement as a solution. This portion of the analysis confirms the place of the

requirements in RDD on its own and along with others.

The second portion of analysis attempts to find for each requirement, its

functionality, features, and facilities and the need for these under different

conditions and constraints. Functionality states “how to achieve the requirement

goal.” Features describe the attributes of functionality, and facilities provide its

delivery, administration, and communication to other systems.

**Process Model of Elicitation and Analysis**

A generic process model of the elicitation and analysis process is shown in

Figure 3.3. Each organization will have its own version or instantiation of this

general model depending on local factors, such as the expertise of the staff, the

type of system being developed, the standards used, etc.

**FIGURE 3.3** Requirements Elicitation and Analysis Process Model

The process activities are:

1. **Domain Understanding.** Analysts must develop their understanding of

the application domain. For example, if a system for a supermarket is

required, the analyst must find out how supermarkets operate.

2. **Requirements Collection.** This is the process of interacting with stakeholders

in the system to discover their requirements. Obviously, domain

understanding develops further during this activity.

3. **Classification.** This activity takes the unstructured collection of requirements

and organizes them into coherent clusters.

INTRODUCTION TO SOFTWARE REQUIREMENTS SPECIFICATION **57**

4. **Conflict Resolution.** Inevitably, where multiple stakeholders are involved,

requirements will conflict. This activity is concerned with findings and

resolving these conflicts.

5. **Prioritization.** In any set of requirements some priorities will be more

important than others. This stage involves interaction with stakeholders

to discover the most important requirements.

6. **Requirements Checking.** The requirements are checked to discover if

they are complete, consistent, and in accordance with what stakeholders

really want from the system.

**3.2.2 Requirements Documentation**

Requirements documentation is a very important activity, which is written after the

requirements elicitation and analysis. This is the way to represent requirements in a

consistent format. The requirements document is called the Software Requirements

Specification (SRS).

The SRS is a specification for a particular software product, program, or set

of programs that perform certain functions in a specific environment. It serves a

number of purposes depending on who is writing it. First, the SRS could be written

by the customer of a system. Second, the SRS could be written by a developer of the

system. The two scenarios create entirely different situations and establish entirely

different purposes for the document. In the first case, the SRS is used to define

the needs and expectations of the users. In the second case, the SRS is written

for a different purpose and serves as a contract document between customer and

developer.

Thus, requirements must be written so they are meaningful not only to the

customers but also to designers on the development team.

**Requirements Definition Document**

The system documentation contains a record of the requirements in the customer’s

terms. This requirements definition document describes what the customer would

like to see.

\_ First, we outline the general purpose of the system. References to other related

systems are included, and we incorporate any terms and abbreviations that

may be useful.

\_ Next, we describe the background and objectives of system development.

For example, if a system is to replace an existing approach, we explain why

the existing system is unsatisfactory. Current methods and procedures are

outlined in enough detail so we can isolate those elements with which the

customer is happy from those that are disappointing.

**58** SOFTWARE ENGINEERING AND TESTING

\_ If the customer has a proposed new approach to solving the problem, we

outline a description of the approach. Remember, though, that the purpose

of the requirements document is to discuss the problem, not the solution; the

focus should be on how the system is to meet the customer’s needs.

\_ Once we record this overview of the problem, we describe the detailed

characteristics of the proposed system. We define the system boundaries and

interfaces across it. The system functions are also explained. Also, we include

a complete list of data elements and classes and their characteristics.

\_ Finally, we discuss the environment in which the system will operate. We

include requirements for support, security, and privacy and any special

hardware or software constraints should be addressed.

**3.2.3 Requirements Review**

A requirements review is a manual process, which involves multiple readers from

both client and contractor staff checking the requirements document for anomalies

and omissions.

A requirements review can be informal or formal.

1. **Informal Review.** Informal reviews simply involve contractors discussing

requirements with as many system stakeholders as possible. It is surprising

how often communication between system developers and stakeholders

ends after elicitation and there is no conformation that the documented

requirements are what the stakeholders really said they wanted.

Many problems can be detected simply by talking about the system to

stakeholders before making a commitment to a formal review.

2. **Formal Review.** In a formal requirements review, the development team

should ‘walk’ the client through the system requirements, explaining the

implications of each requirement. The review team should check each

requirement for consistency and should check the requirements as a whole

for completeness. Reviewers may also check for:

(*a*) *Verifiability*: are the requirements as stated realistically testable?

(*b*) *Comprehensibility*: is the requirement property understood by the

procurers or end-users of the system?

(*c*) *Traceability*: is the origin of the requirement clearly stated? You may

have to go back to the source of the requirement to assess the impact

of a change. Traceability is important as it allows the impact of change

on the rest of the system to be assessed. We discuss it in more detail in

the following section.

(*d*) *Adaptability*: is the requirement adaptable? That is, can the requirement

be changed without large-scale effects on other system requirements?

INTRODUCTION TO SOFTWARE REQUIREMENTS SPECIFICATION **59**

Conflicts, contradictions, errors, and omissions in the requirements should be

pointed out during the review and formally recorded. It is then up to the users,

the system procurer, and the system developer to negotiate a solution to these

identified problems.

**3.2.4 Requirements Management**

Requirements define the capability that the software system solution must deliver

and the intended results that must result on its application to business problems. In

order to generate such requirements, a systematic approach is necessary, through a

formal management process called Requirements Management.

Requirements management is defined as a systematic approach to eliciting,

organizing, and documenting the requirements of the system, and a process that

establishes and maintains agreement between the customer and project team on

the changing requirements of the system.

**Classes of Requirements Management**

From an evolution perspective, requirements fall into two classes:

1. *Enduring requirements.* These are relatively stable requirements which derive

from the core activity of the organization and which relate directly to the

domain of the system. For example, in a hospital there will always be

requirements concerned with patients, doctors, nurses, treatments, etc. These

requirements may be derived from domain models that show the entities and

relations which characterize an application domain.

**FIGURE 3.4** Requirements Evolution

2. *Volatile requirements.* These are requirements which are likely to change during

the system development or after the system has been put into operation.

Examples of volatile requirements are requirements resulting from government

health-care policies.

**60** SOFTWARE ENGINEERING AND TESTING

**Requirements Management Planning**

Planning is an essential first stage in the requirements management process.

Requirements management is very expensive and, for each project, the planning

stage establishes the level of requirements management detail required. During

the requirements management stage, you have to decide on:

1. *Requirements identification*. Each requirement must be uniquely identified so

that it can be cross-referred by other requirements and so that it may be used

in traceability assessments.

2. *A change management process*. This is the set of activities that assess the impact

and cost of changes. We discuss it in more detail in the following section.

3. *Traceability policies*. These policies define the relationships between requirements

and between the requirements and the system design that should be recorded

and how these records should be maintained.

4. *CASE tool support*. Requirements management involves the processing of large

amounts of information about the requirements. Tools, which may be used,

range from specialist requirements management system to spreadsheets and

simple database systems.

The process steps and their outputs which when implemented will lead to the

most acceptable RDD and SRS through requirements management are given in

Table 3.1.

**TABLE 3.1 Process Steps for Requirements Management**

**Steps Outputs**

Study the domain Domain knowledge improved for better

solutions.

Analyze the problems Increased understanding of the problem.

Understand user needs Real and genuine needs that need solving

identified.

Determine and define the system

and system scope

First prototype system model and its scope

definition.

Manage scope System scope: Feasible and deliverable.

Refine and evaluate the system

definition

Broader system scope defined in terms of

deliverables.

Build the right system A system accepted by the users.