UNIT - II

SOFTWARE REQUIREMENTS & REQUIREMENTS ENGINEERING PROCESS

Part I

Software Requirements:

1. Functional and non-functional requirements
2. user requirements
3. system requirements
4. interface specification
5. the software requirements document

Part II

Requirementsengineering process:

1. Feasibility studies
2. Requirements elicitation and analysis
3. Requirements validation
4. Requirements management

Part I

Part I

Software Requirements:

1. Functional and non-functional requirements
2. user requirements
3. system requirements
4. interface specification
5. the software requirements document

1. Functional and non-functional requirements

Functional and Non-functional Requirements:

Software system requirements are often classified as functional requirements ornonfunctional requirements:

1. ***Functional requirements:*** These are statements of services the system should provide, how the system should react to particular inputs, and how the system should behave in particular situations.

In some cases, the functional requirements may also explicitly state what the system should not do.

2. ***Non-functional requirements****:* These are constraints on the services or functions offered by the system.

They include timing constraints, constraints on the development process, and constraints imposed by standards.

Non-functional requirements often apply to the system as a whole, ratherthan individual system features or services.

Functional requirements:

The functional requirements for a system describe what the system should do. These requirements depend on

- the type of software being developed,
- the expected users of the software, and
- The general approach taken by the organization when writing requirements.

When expressed as user requirements, functional requirements are usually described in an abstract way that can be understood by system users. However, more specific functional system requirements describe the system functions, its inputs and outputs, exceptions, etc., in detail.

Functional system requirements vary from general requirements coveringwhat the system should do to very specific requirements reflecting local ways of working or an organization's existing systems.

Example:

Functional requirements for the MHC-PMS system, used to maintain information about patients receiving treatment for mental health problems:

1. A user shall be able to search the appointments lists for all clinics.
2. The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.
3. Each staff member using the system shall be uniquely identified by his or her eight-digit employee number.

In principle, the functional requirements specification of a system should be both complete and consistent.

- Completeness means that all services required by the user should be defined.
- Consistency means that requirements should not have contradictory definitions.

In practice, for large, complex systems, it is practically impossible to achieve requirements consistency and completeness.

One reason for this is that it is easy to make mistakes and omissions when writing specifications for complex systems.

Another reason is that there are many stakeholders in a large system. A stakeholder is a person or role that is affected by the system in some way. Stakeholders have different and often inconsistent needs. These inconsistencies may not be obvious when the requirements are first specified, so inconsistent requirements are included in the specification.

The problems may only emerge after deeper analysis or after the system has been delivered to the customer.

Non-functional requirements:

Non-functional requirements are the requirements that are not directly concerned with the specific services delivered by the system to the users.

They may relate to emerge system properties such as reliability, response time, and store occupancy. Alternatively, they may define constraints on the system implementation such as the capabilities of I/O devices or the data representations used in interfaces with other systems.

Non-functional requirements are often more critical than individual functional requirements. However, failing to meet a non-functional requirement can mean that the whole system is unusable. For example, if an aircraft system does not meet its reliability requirements, it will not be certified as safe for operation.

Although it is often possible to identify which system components implement specific functional requirements (e.g., there may be formatting components that implement reporting requirements), it is often more difficult to relate components to non-functional requirements. The implementation of these requirements may be diffused throughout the system. There are two reasons for this:

1. Non-functional requirements may affect the overall architecture of a system rather than the individual components. For example, to ensure that performance requirements are met, you may have to organize the system to minimize communications between components.

2. A single non-functional requirement, such as a security requirement, may generate a number of related functional requirements that define new system services that are required. In addition, it may also generate requirements that restrict existing requirements.

Classification of non-functional requirements:

Non-functional requirements arise through user needs, because of budget constraints, organizational policies, the need for interoperability with other softwareor hardware systems, or external factors such as safety regulations or privacy legislation.

Following figure shows the classification of non-functional requirements.

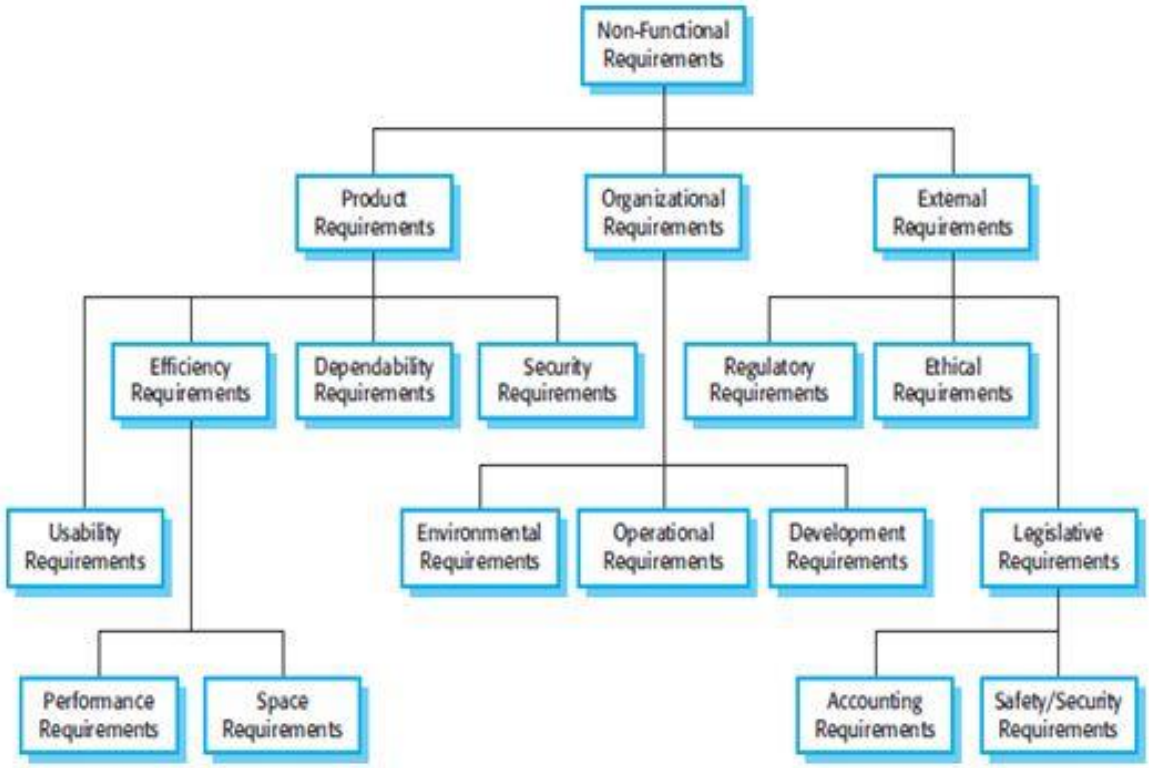


Figure: Types of Non-functional Requirements

From this diagram we can see that the non-functional requirements may come from required characteristics of the software (product requirements), the organization developing the software (organizational requirements), or from external sources

1. ***Product requirements:*** These requirements specify or constrain the behavior of the software.

Examples include performance requirements on how fast the system must execute and how much memory it requires, reliability requirements that set out the acceptable failure rate, security requirements, and usability requirements.

2. ***Organizational requirements:*** These requirements are broad system requirements derived from policies and procedures in the customer's and developer's organization.

Examples include operational process requirements that define how the system will be used, development process requirements that specify the programming language, the development environment or process standards to be used, and environmental requirements that specify the operating environment of the system.

3. ***External requirements:*** This broad heading covers all requirements that arederived from factors external to the system and its development process.

These may include regulatory requirements that set out what must be done for the system to be approved for use by a regulator, such as a central bank; legislative requirements that must be followed to ensure that the system operates within the law; and ethical requirements that ensure that the system will beacceptable to its users and the general public.

Example:

PRODUCT REQUIREMENT
The MHC-PMS shall be available to all dinics during normal working hours (Mon–Fri, 08.30–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.

ORGANIZATIONAL REQUIREMENT
Users of the MHC-PMS system shall authenticate themselves using their health authority identity card.

EXTERNAL REQUIREMENT
The system shall implement patient privacy provisions as set out in HStan-03-2006-priv.

Whenever possible, we should write non-functional requirements quantitatively so that they can be objectively tested. Following figure shows metrics that you can use to specify non-functional system properties. We can measure these characteristics when the system is being tested to check whether or not thesystem has met its nonfunctional requirements.

Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

Figure: Metrics for specifying non-functional requirements

Non-functional requirements such as reliability, safety, and confidentiality requirements are particularly important for critical systems.

Functional Requirements	Non-Functional Requirements
1. A functional requirement defines a system or its component.	A non-functional requirement defines the quality attribute of a software system.
2. It specifies "What should the software system do?"	It places constraints on "How should the software system fulfil the functional requirements?"
3. Functional requirement is specified by User.	Non-functional requirement is specified by technical peoples e.g. Architect, Technical leaders and software developers.
4. It is mandatory.	It is not mandatory.
5. It is captured in use case.	It is captured as a quality attribute.
6. Defined at a component level.	Applied to a system as a whole.
7. Helps you verify the functionality of the software.	Helps you to verify the performance of the software.
8. Functional Testing like System, Integration, End to End, API testing, etc are done.	Non-Functional Testing like Performance, Stress, Usability, Security testing, etc are done.
9. Usually easy to define.	Usually more difficult to define.
Example **1)** Authentication of user whenever he/she logs into the system. **2)** System shutdown in case of a cyber attack. **3)** A Verification email is sent to user whenever he/she registers for the first time on some software system.	**Example** **1)** Emails should be sent with a latency of no greater than 12 hours from such an activity. **2)** The processing of each request should be done within 10 seconds **3)** The site should load in 3 seconds when the number of simultaneous users are > 10000

User requirements:

These requirements describe what the end-user wants from the software system. User requirements are usually expressed in natural language and are typically gathered through interviews, surveys, or user feedback.

1. User requirements are written for customers
2. They are usually expresses in natural language.
3. Because of this, they are easy to understand
4. They describe services and features provided by system
5. This may include diagrams and tables which are understood by system users
6. The system users do not need technical knowledge to understand these
7. User requirements are for client managers, system end users, client engineers, contractor managers and system architects
8. They are gathered through various means such as interviews, surveys, or user feedback.

Example:

User Requirement Definition

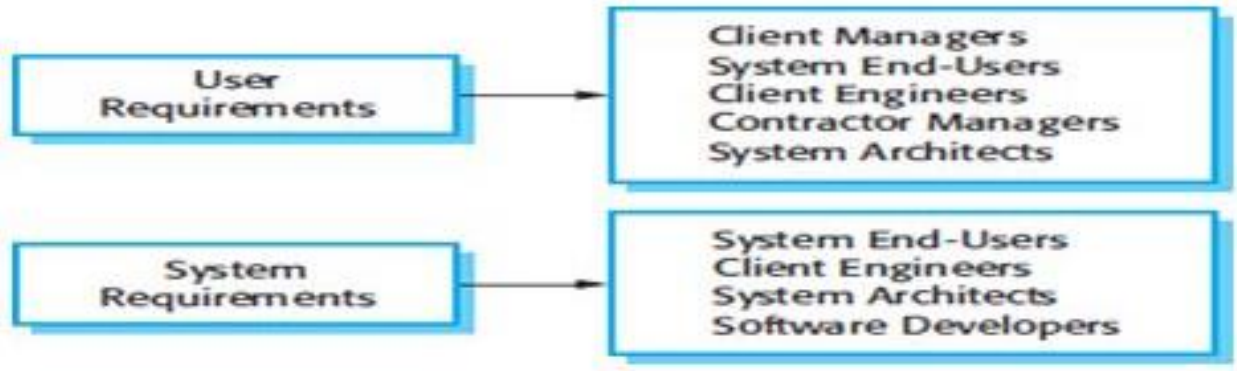
1. The MHC-PMS shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System Requirements Specification

1.1 On the last working day of each month, a summary of the drugs prescribed, their cost, and the prescribing clinics shall be generated.
1.2 The system shall automatically generate the report for printing after 17.30 on the last working day of the month.
1.3 A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed, and the total cost of the prescribed drugs.
1.4 If drugs are available in different dose units (e.g., 10 mg, 20 mg) separate reports shall be created for each dose unit.
1.5 Access to all cost reports shall be restricted to authorized users listed on a management access control list.

This example from a mental health care patient management system (MHC- PMS) shows how a user requirement may be expanded into several system requirements.

Different levels of requirements are useful because they communicate information about the system to different types of reader.



System requirements:

These requirements specify the technical characteristics of the software system, such as its architecture, hardware requirements, software components, and interfaces. System requirements are typically expressed in technical terms and are often used as a basis for system design.

Salient features

1. Written for implementation team
2. They are written in technical language / technical terms
3. System Requirements describe the detailed description of services, features and complete operations of system
4. System Requirements may include system models and system designs
5. System Requirements can be understood by implementation team with technical knowledge.
6. System Requirements are for architects, software Developers, client engineers, system users and overall implementation team
7. They form basis for a system design

Interface Specification:

What is Interface?

- A point where two systems, subjects, organizations, etc. meet and interact.
- A device or program enabling a user to communicate with a computer.
- A inter face is a intersection between system and environment.
- Interface =system /environment

What is specification?

- A Specification is a agreement Between the produce of the services Consumer of that services

What is Interface specification?

- All software systems must operate with existing systems that have already been implemented and installed in an environment.
- If the new system and existing systems must work together, the interfaces of existing systems have to be precisely specified.
- These specifications should be defined early in the process and included in the requirements document.

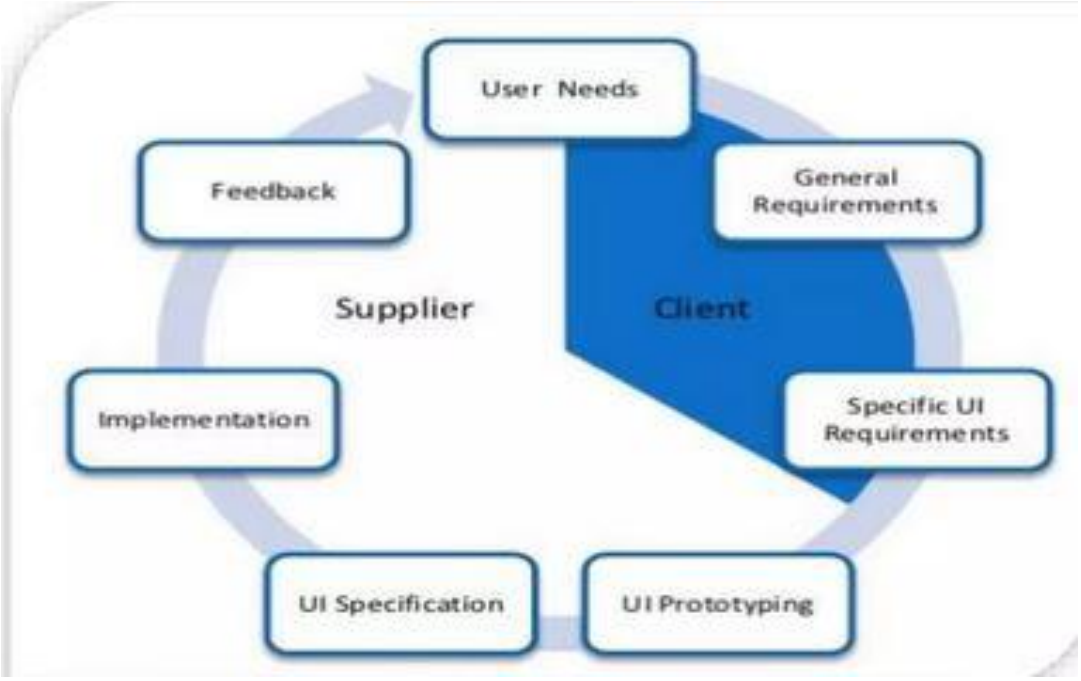
Types of Interface Specification:

- **Procedural interfaces.**
- **Data structures.**
- **Representations of data.**

Procedural interfaces where existing programs or sub-systems offer a range of services that are accessed by calling interface procedures. In simple words it Is used for calling the existing programs by the new programs These interfaces are sometimes called Application Programming Interfaces (APLs).

- **Data structures** that are passed from one sub-system to another. Graphical data models are the best notations for this type of description

- **Representations of data** (such as the ordering of bits) that have been established for an existing sub-system. These interfaces are most common in embedded, real-time system. Some programming languages such as Ada (although not Java) support this level of Specification.
- Sub system requesting service from other sub systems.



For 5 marks question

Interface Specification:

Most systems must operate with other systems and the operating interfaces mustbe specified as part of the requirements.

Three types of interface may have to be defined

- ***Procedural interfaces*** where existing programs or sub-systems offer a range of services that are accessed by calling interface procedures. These interfaces are sometimes called Application Programming Interfaces (APIs)
- ***Data structures that are exchanged*** that are passed from one sub-systemto another. Graphical data models are the best notations for this type of description
- ***Data representations*** that have been established for an existing sub-

system Formal notations are an effective technique for interface specification.

oOo

Software Requirements Document:

The software requirements document (sometimes called the software requirements specification or SRS) is an official statement of what the system developers should implement.

It should include both the user requirements for a system and a detailed specification of the system requirements.

Sometimes, the user and system requirements are integrated into a single description. In other cases, the user requirements are defined in an introduction to the system requirements specification. If there are a large number of requirements, the detailed system requirements may be presented in a separate document.

Requirements documents are essential when an outside contractor is developing the software system. The requirements document has a diverse set of users, ranging from the senior management of the organization that is paying for the system to the engineers responsible for developing the software. Following figure shows possible users of the document and how they use it.

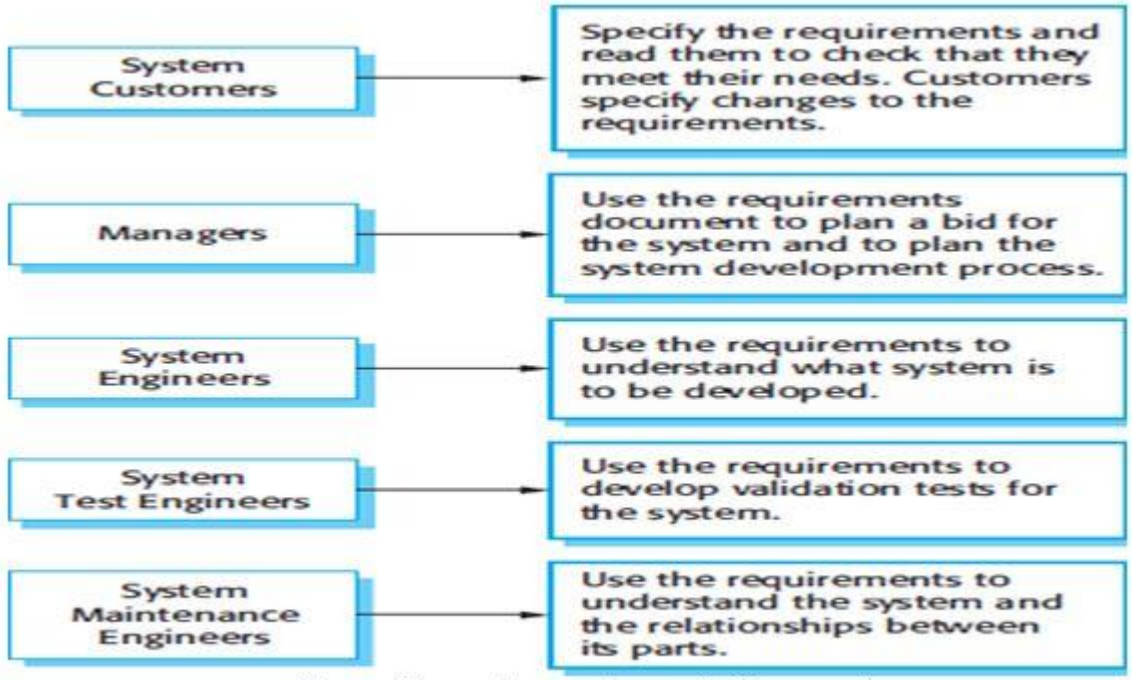


Figure: Users of a requirements document

The image in the next page shows one possible organization for a requirements document that is based on an IEEE standard for requirements documents (IEEE, 1998). This standard is a generic standard that can be adapted to specific uses.

Chapter	Description
Preface	This should define the expected readership of the document and describe its version history, including a rationale for the creation of a new version and a summary of the changes made in each version.
Introduction	This should describe the need for the system. It should briefly describe the system's functions and explain how it will work with other systems. It should also describe how the system fits into the overall business or strategic objectives of the organization commissioning the software.
Glossary	This should define the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.
User requirements definition	Here, you describe the services provided for the user. The non-functional system requirements should also be described in this section. This description may use natural language, diagrams, or other notations that are understandable to customers. Product and process standards that must be followed should be specified.
System architecture	This chapter should present a high-level overview of the anticipated system architecture, showing the distribution of functions across system modules. Architectural components that are reused should be highlighted.
System requirements specification	This should describe the functional and non-functional requirements in more detail. If necessary, further detail may also be added to the non-functional requirements. Interfaces to other systems may be defined.
System models	This might include graphical system models showing the relationships between the system components, the system, and its environment. Examples of possible models are object models, data-flow models, or semantic data models.
System evolution	This should describe the fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on. This section is useful for system designers as it may help them avoid design decisions that would constrain likely future changes to the system.
Appendices	These should provide detailed, specific information that is related to the application being developed; for example, hardware and database descriptions. Hardware requirements define the minimal and optimal configurations for the system. Database requirements define the logical organization of the data used by the system and the relationships between data.
Index	Several indexes to the document may be included. As well as a normal alphabetic index, there may be an index of diagrams, an index of functions, and so on.

In this case, they have extended the standard to include information about predicted system evolution. This information helps the maintainers of the system and allows designers to include support for future system features.

Requirements specification:

Requirements specification is the process of writing down the user and system requirements in a requirements document.

Ideally, the user and system requirements should be clear, unambiguous, easy to understand, complete, and consistent. In practice, this is difficult to achieve as stakeholders interpret the requirements in different ways and there are often inherent conflicts and inconsistencies in the requirements.

The user requirements for a system should describe the functional and nonfunctional requirements so that they are understandable by system users who don't have detailed technical knowledge. Ideally, they should specify only the external behavior of the system. The requirements document should not include details of the system architecture or design. Consequently, if you are writing user requirements, you should not use software jargon, structured notations, or formal notations. You should write user requirements in natural language, with simple tables, forms, and intuitive diagrams.

System requirements are expanded versions of the user requirements that are used by software engineers as the starting point for the system design. They add detail and explain how the user requirements should be provided by the system. They may be used as part of the contract for the implementation of the system and should therefore be a complete and detailed specification of the whole system. Ideally, the system requirements should simply describe the external behavior of the system and its operational constraints. They should not be concerned with howthe system should be designed or implemented.

User requirements are almost always written in natural language supplemented by appropriate diagrams and tables in the requirements document. System requirements may also be written in natural language but other notations based on forms, graphical system models, or mathematical system models can also be used. Following figure summarizes the possible notations that could be used for writing system requirements.

Notation	Description
Natural language sentences	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
Design description languages	This approach uses a language like a programming language, but with more abstract features to specify the requirements by defining an operational model of the system. This approach is now rarely used although it can be useful for interface specifications.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system; UML use case and sequence diagrams are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification. They cannot check that it represents what they want and are reluctant to accept it as a system contract.

Ways of writing system requirements specification

oOo

Part II

Part II

Requirementsengineering process:

1. Feasibility studies
2. Requirements elicitation and analysis
3. Requirements validation
4. Requirements management

Requirements Engineering Process:

The requirements engineering process aims to produce an agreed requirements document that specifies a system satisfying stakeholder requirements.

Requirements are usually presented at two levels of detail. End-users and customers need a high-level statement of the requirements; system developers needa more detailed system specification.

There are four main activities in the requirements engineering process:

1. ***Feasibility study*** (These focus on assessing if the system is useful to thebusiness),
2. ***Requirements elicitation and analysis*** (discovering requirements),
3. ***Requirements specification*** (Converting these requirements i nto some standard form), and
4. ***Requirements validation*** (checking that the requirements actually define thesystem that the customer wants).

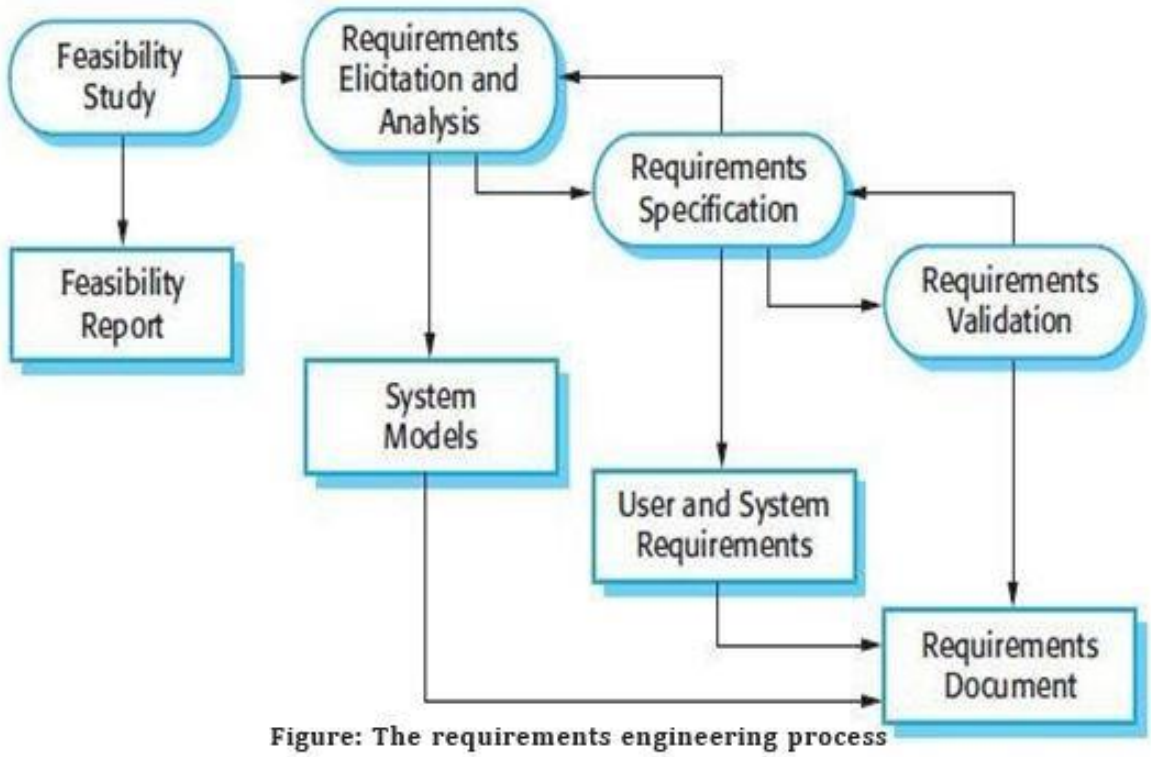


Figure: The requirements engineering process

Feasibility study:

An estimate is made of whether the identified user needs may be satisfied using current software and hardware technologies.

The study considers whether the proposed system will be cost-effective from a business point of view and if it can be developed within existing budgetary constraints.

A feasibility study should be relatively cheap and quick.

The result should inform the decision of whether or not to go ahead with a more detailed analysis.

- A feasibility study is a short, focused study that should take place early in the RE process.
- It should answer three key questions:
  a) does the system contribute to the overall objectives of the organization?
  b) can the system be implemented within schedule and budget using current technology? And
  c) can the system be integrated with other systems that are used?

  If the answer to any of these questions is no, you should probably not goahead with the project.

Requirements elicitation and analysis:

After an initial feasibility study, the next stage of the requirements engineering process is requirements elicitation and analysis.

In this activity, software engineers work with customers and system end-users to find out about the application domain, what services the system should provide, the required performance of the system, hardware constraints, and so on.

Requirements elicitation and analysis may involve a variety of different kinds of people in an organization.

A system stakeholder is anyone who should have some direct or indirect influence on the system requirements. Stakeholders include end users who will interact with the system and anyone else in an organization who will be affected byit. Other system stakeholders might be engineers who are developing or maintaining other related systems, business managers, domain experts, and trade union representatives.

A process model of the elicitation and analysis process is shown in Figure below.

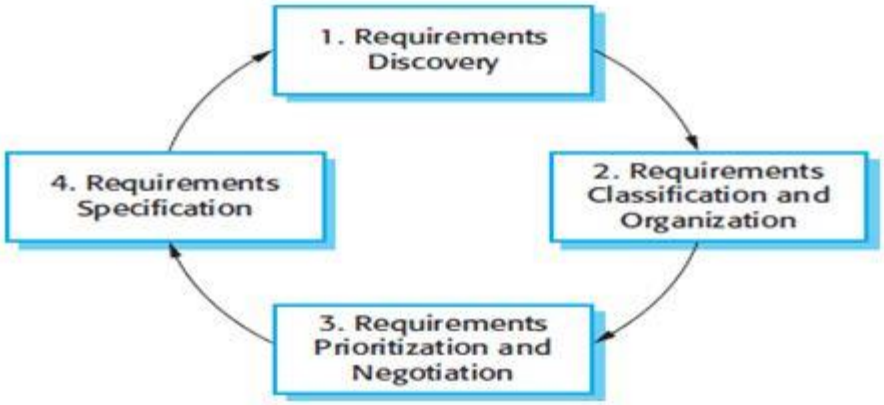


Figure: The requirements elicitation and analysis process

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.

The process activities are:

1. Requirements discovery:

   This is the process of interacting with stakeholders of the system to discover their requirements. Domain requirements from stakeholders and documentation are also discovered during this activity.

2. Requirements classification and organization:

   This activity takes the unstructured collection of requirements, groups related requirements, and organizes them into coherent clusters. The most common way of grouping requirements is to use a model of the system architecture to identify sub-systems and to associate requirements with each sub-system. In practice, requirements engineering and architectural design cannot be completely separate activities.

3. Requirements prioritization and negotiation:

   Inevitably, when multiple stakeholders are involved, requirements will conflict. This activity is concerned with prioritizing requirements and finding andresolving requirements conflicts through negotiation. Usually, stakeholders have to meet to resolve differences and agree on compromise requirements.

4. Requirements specification:

   The requirements are documented and input into the next round of the spiral. Formal or informal requirements documents may be produced.

The above figure shows that requirements elicitation and analysis is an iterative process with continual feedback from each activity to other activities.

The process cycle starts with requirements discovery and ends with the requirements documentation. The analyst's understanding of the requirements improves with each round of the cycle. The cycle ends when the requirements document is complete.

Eliciting and understanding requirements from system stakeholders is a difficult process for several reasons:

1. Stakeholders often don't know what they want from a computer system except in the most general terms; they may find it difficult to articulate what they want the system to do; they may make unrealistic demands because they don't know what is and isn't feasible.
2. Stakeholders in a system naturally express requirements in their own terms and with implicit knowledge of their own work. Requirements engineers, without experience in the customer's domain, may not understand these requirements.
3. Different stakeholders have different requirements and they may express these in different ways. Requirements engineers have to discover all potential sources of requirements and discover commonalities and conflict.
4. Political factors may influence the requirements of a system. Managers may demand specific system requirements because these will allow them to increase their influence in the organization.
5. The economic and business environment in which the analysis takes place is dynamic. It inevitably changes during the analysis process. The importance of particular requirements may change. New requirements may emerge from new stakeholders who were not originally consulted.

Requirements specification:

Requirements specification is the activity of translating the information gathered during the analysis activity into a document that defines a set of requirements. Two types of requirements may be included in this document. User requirements are abstract statements of the system requirements for the customer and end-user of the system; system requirements are a more detailed description of the functionality to be provided.

Requirements validation:

Requirements validation is the process of checking that requirements actually define the system that the customer really wants.

It overlaps with analysis as it is concerned with finding problems with the requirements.

Requirements validation is important because errors in a requirements document can lead to extensive rework costs when these problems are discovered during development or after the system is in service.

The cost of fixing a requirements problem by making a system change is usually much greater than repairing design or coding errors.

During the requirements validation process, different types of checks should be carried out on the requirements in the requirements document. These checks include:

1. Validity checks:

   A user may think that a system is needed to perform certain functions. However, further thought and analysis may identify additional or different functions that are required. Systems have diverse stakeholders with different needs and any set of requirements is inevitably a compromise across the stakeholder community.

2. Consistency checks:

   Requirements in the document should not conflict. That is, there should not be contradictory constraints or different descriptions of the same system function.

3. Completeness checks:

   The requirements document should include requirements that define all functions and the constraints intended by the system user.

4. Realism checks:

   Using knowledge of existing technology, the requirements should be checked to ensure that they can actually be implemented. These checks should also take account of the budget and schedule for the system development.

5. Verifiability:

   To reduce the potential for dispute between customer and contractor, system requirements should always be written so that they are verifiable. This means that you should be able to write a set of tests that can demonstrate that the delivered system meets each specified requirement.

There are a number of requirements validation techniques that can be usedindividually or in conjunction with one another:

1. Requirements reviews:

The requirements are analyzed systematically by a team of reviewers who check for errors and inconsistencies.

2. Prototyping:

In this approach to validation, an executable model of the system in questionis demonstrated to end-users and customers. They can experiment with this modelto see if it meets their real needs.

3. Test-case generation:

Requirements should be testable. If the tests for the requirements are devised as part of the validation process, this often reveals requirements problems. If a testis difficult or impossible to design, this usually means that the requirements will be difficult to implement and should be reconsidered. Developing tests from the user requirements before any code is written is an integral part of extreme programming.

Requirements management:

Once a system has been installed and is regularly used, new requirements inevitably emerge due to business, organizational, and technical changes which lead to changes to the requirements for a software system.

Requirements management is the process of understanding and controlling changes to system requirements.

You need to establish a formal process for making change proposals andlinking these to system requirements.

The formal process of requirements management should start as soon as a draft version of the requirements document is available.

However, you should start planning how to manage changing requirements during the requirements elicitation process.

Requirements management planning:

Planning is an essential first stage in the requirements management process. Itestablishes the level of requirements management detail that is required.

During this requirements management stage, you have to decide on:

1. Requirements identification:

   Each requirement must be uniquely identified so that it can be cross- referenced with other requirements and used in traceability assessments.

2. A change management process:

   This is the set of activities that assess the impact and cost of changes.

3. Traceability policies:

   These policies define the relationships between each requirement and between the requirements and the system design that should be recorded. The traceability policy should also define how these records should be maintained.

4. Tool support:

   Requirements management involves the processing of large amounts of information about the requirements. Tools that may be used range from

specialist requirements management systems to spreadsheets and simple database systems.

Requirements management needs automated support and the software tools for thisshould be chosen during the planning phase. You need tool support for:

1. Requirements storage:

The requirements should be maintained in a secure, managed data store thatis accessible to everyone involved in the requirements engineering process.

2. Change management:

The process of change management is simplified if active tool support isavailable.

3. Traceability management:

Tool support for traceability allows related requirements to be discovered. Some tools are available which use natural language processing techniques to help discover possible relationships between requirements.

For small systems, it may not be necessary to use specialized requirements management tools. It may be supported using the facilities available in word processors, spreadsheets, and PC databases. However, for larger systems, more specialized tool support is required.

Requirements change management:

Requirements change management should be applied to all proposed changes to a system's requirements after the requirements document has been approved.

Change management is essential because you need to decide if the benefits of implementing new requirements are justified by the costs of implementation.

The advantage of using a formal process for change management is that all change proposals are treated consistently and changes to the requirements document are made in a controlled way.

Figure: Requirements change management

There are three principal stages to a change management process:

1. Problem analysis and change specification:

   - The process starts with an identified requirements problem or, sometimes, with a specific change proposal.
   - During this stage, the problem or the change proposal is analyzed to check that it is valid. This analysis is fed back to the change requestor who may respond with a more specific requirements change proposal, or decide to withdraw the request.

2. Change analysis and costing:

   - The effect of the proposed change is assessed using traceability information and general knowledge of the system requirements.
   - The cost of making the change is estimated both in terms of modifications to the requirements document and, if appropriate, to the system design andimplementation.
   - Once this analysis is completed, a decision is made whether or not to proceed with the requirements change.

3. Change implementation:

   - The requirements document and, where necessary, the system design and implementation, are modified.
   - We should organize the requirements document so that we can make changes to it without extensive rewriting or reorganization.
   - As with programs, changeability in documents is achieved by minimizing external references and making the document sections as modular as possible. Thus, individual sections can be changed and replaced without affecting other parts of the document.

If a new requirement has to be urgently implemented, there is always a temptation to change the system and then retrospectively modify the requirements document.

oOo