# Introductions

Requirements →the descriptions of what the system should do, the services that it provides and the constraints on its operation.

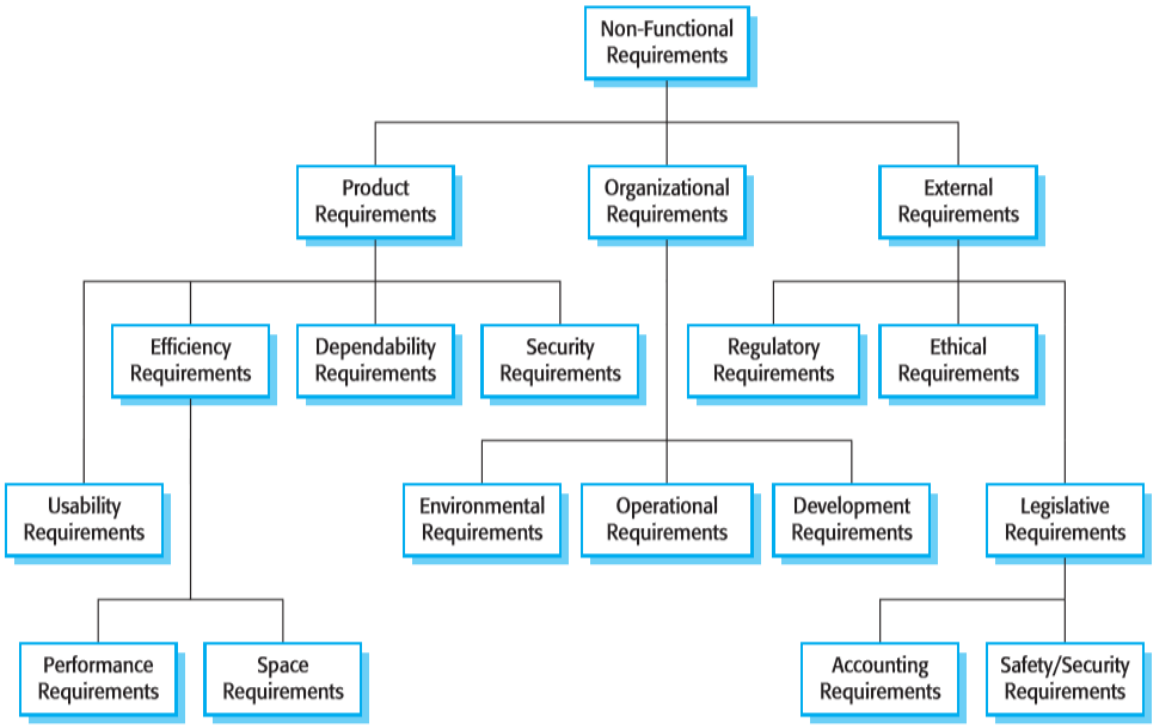
Requirements engineering (RE)→ the process of finding out, analyzing, documenting and checking these services and constraints.

Software system requirements are often classified as:

## Functional requirements or non-functional 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, rather than individual system features or services.



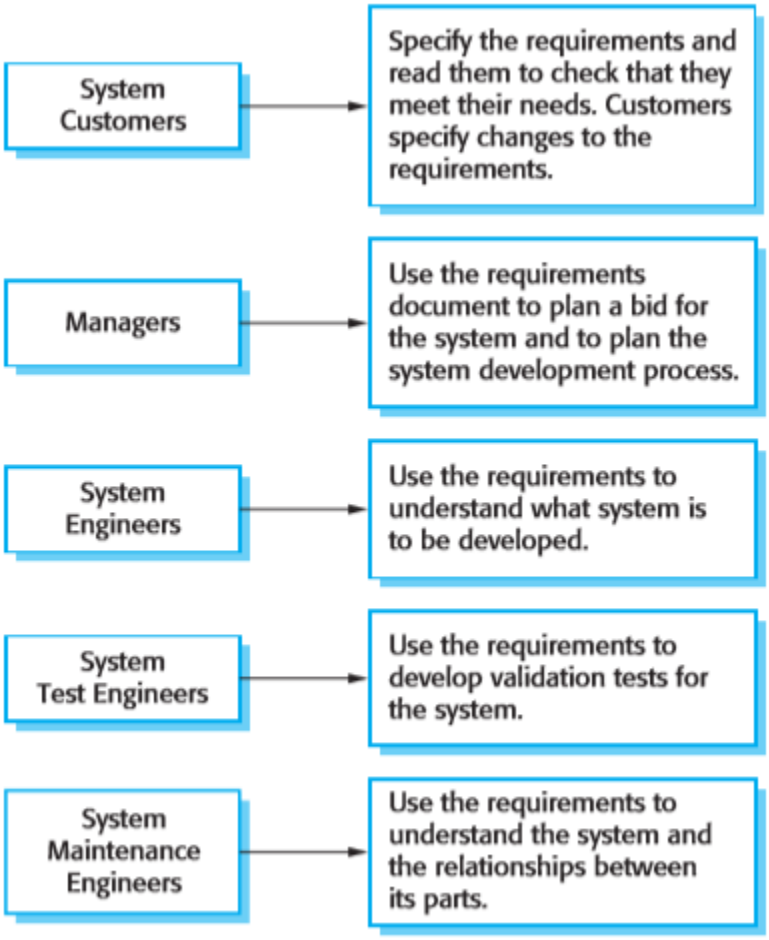
**Metrics for specifying non-functional requirements**

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

## The software requirements document

- sometimes called the software requirements specification or SRS
- An official statement of what the system developers should implement.
- It should include
    - 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.
- However, agile development methods argue that requirements change so rapidly that a requirements document is out of date as soon as it is written, so the effort is largely wasted.
- In Extreme Programming there is no formal requirement instead, collect user requirements incrementally and write these on cards as user stories.
- The user then prioritizes requirements for implementation in the next increment of the system.

## Users of a requirements document



## The structure of a requirements document

| 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. |

## Requirements specification

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, there are often inherent conflicts and inconsistencies in the requirements as stakeholders interpret the requirements in different ways.

## Ways of writing a system requirements specification

| 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. |

# Requirements engineering process

Requirements engineering processes may include four high-level activities.

- Assessing if the system is useful to the business (feasibility study),
- Discovering requirements (elicitation and analysis),
- Converting these requirements into some standard form (specification), and
- Checking that the requirements actually define the system that the customer wants (validation).

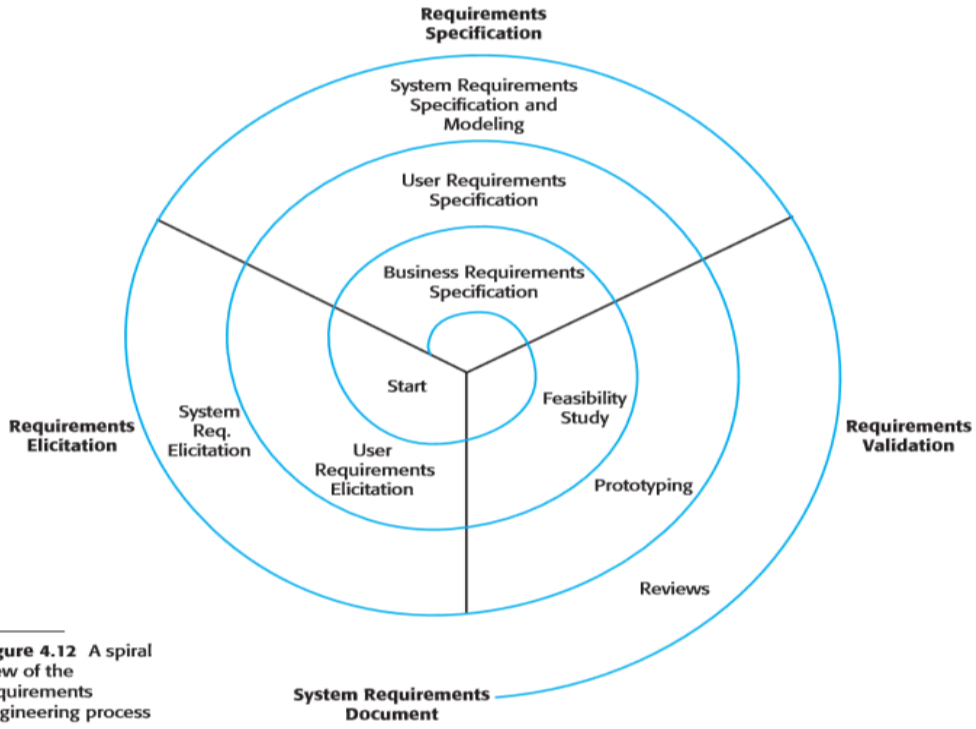
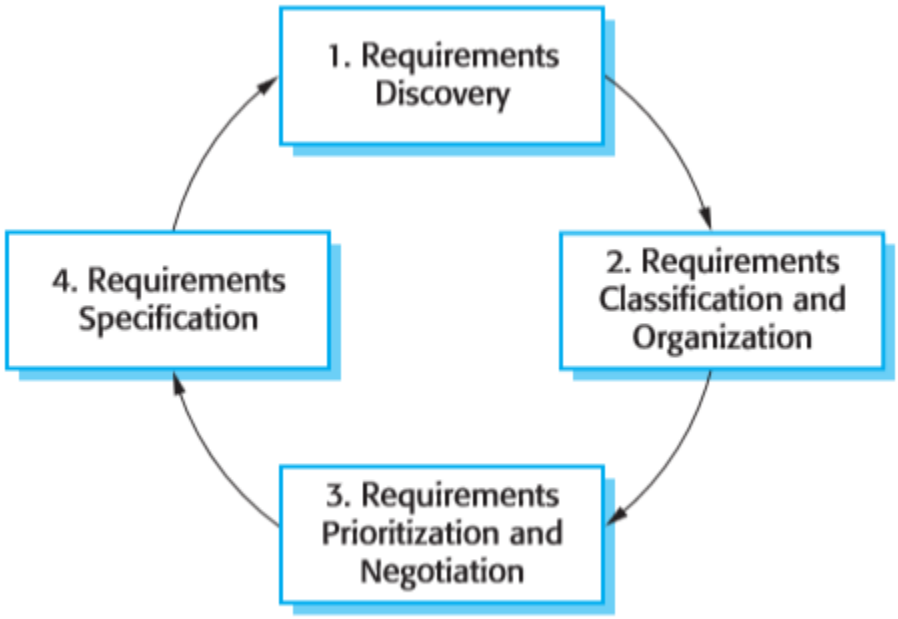



**Figure 4.12** A spiral view of the requirements engineering process

## Requirements elicitation and analysis

Software engineers work with a range of system stakeholders to find out about the application domain, the services that the system should provide, the required system performance, hardware constraints, other systems, etc.

## The requirements elicitation and analysis process



**Requirements discovery**

Interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage.

**Requirements classification and organization**

Groups related requirements and organizes them into coherent clusters.

**Prioritization and negotiation**

Prioritizing requirements and resolving requirements conflicts.

**Requirements specification**

Requirements are documented and input into the next round of the spiral.

Why is it a difficult process to Eliciting and understanding requirements from system stakeholders? Explain

## Requirements validation

Requirements validation is concerned with demonstrating that the requirements define the system that the customer really wants. Requirements error costs are high so validation is very important.

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

- **Validity**: does the system provide the functions which best support the customer's needs?
- **Consistency**: are there any requirements conflicts?
- **Completeness**: are all functions required by the customer included?
- **Realism**: can the requirements be implemented given available budget and technology?
- **Verifiability**: can the requirements be checked?

### Requirements validation techniques

- **Requirements reviews**

  Systematic manual analysis of the requirements. Regular reviews should be held while the requirements definition is being formulated. What to look for:

  ➔ **Verifiability**: is the requirement realistically testable?
  ➔ **Comprehensibility**: is the requirement properly understood?
  ➔ **Traceability**: is the origin of the requirement clearly stated?
  ➔ **Adaptability**: can the requirement be changed without a large impact on other requirements?

- **Prototyping**

Using an executable model of the system to check requirements.

- **Test-case generation**

Developing tests for requirements to check testability.

## Requirements Management

Requirements management is the process of managing changing requirements during the requirements engineering process and system development. New requirements emerge as a system is being developed and after it has gone into use. Reasons why requirements change after the system's deployment:

1. The business and technical environment of the system always changes after installation.
2. The people who pay for a system and the users of that system are rarely the same people.
3. Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory.

## Requirements management planning

Planning is an essential first stage in the requirements management process. The planning stage establishes the level of requirements management detail that is required. During the requirements management stage, you have to decide on:

1. **Requirements identification**
   Each requirement must be uniquely identified.
2. **A change management process**
   set of activities that assess the impact and cost of changes.
3. **Traceability policies**
   define the relationships between each requirement and between the requirements and the system design that should be recorded.
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 change management


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.

2. **Change analysis and costing**

   The cost of making the change is estimated both in terms of modifications to the requirements document and to the system design and implementation. 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