INSTITUTO SUPERIOR DE TECNOLOGIAS AVANÇADAS DE LISBOA

Software Engineering

João Caldeira

Invited Professor

Email. joaocarlos.caldeira@my.istec.pt

Mob. +351 917769544

November 23, 2022

INSTITUTO SUPERIOR DE TECNOLOGIAS AVANÇADAS DE LISBOA

Requirements Engineering

November 23, 2022

Requirements Engineering

What are the Requirements ?

- A **software project starts** when someone gets an **idea or a demand** presents itself
  - new company/organization
  - new department
  - new product
  - new rules and regulations
  - new business processes
- The requirements **state what** a system **should do** and **under what constraints**
- **Requirements** are a **fundamental** part of the **communication** between the client and the software engineering team
  - Often, they are part of a legal contract signed

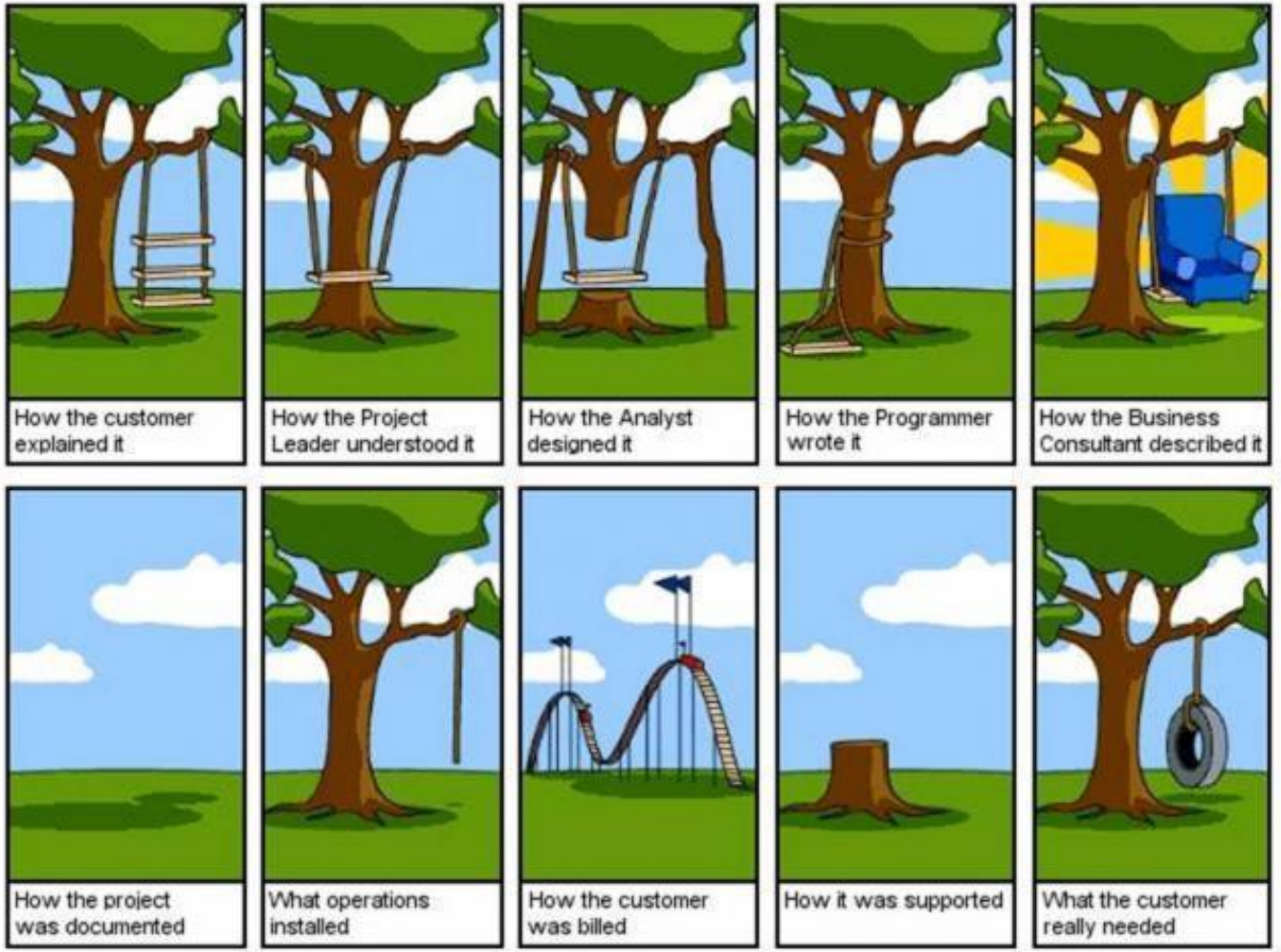
Requirements Engineering

Definition in SWEBOK

- .....a **software requirement is** a **property** which must be **exhibited** in order **to solve** some **problem** in the real world.....

Requirements Engineering

Why are they needed ?



Requirements Engineering

Topics covered

- **Product** and **process** requirements
  - **Functional** and **non-functional** requirements
- **User** requirements
- **System** requirements
- The software **requirements document**

Requirements Engineering

Types of requirement

- **User requirements**
  - Statements in **natural language & diagrams** of the **services** the system provides & its operational **constraints**
  - Written for customers

- **System requirements**
  - A structured document setting out **detailed descriptions of system's functions, services** & operational **constraints**
  - Defines what should be implemented, **may be part of a contract between** client and contractor

Requirements Engineering

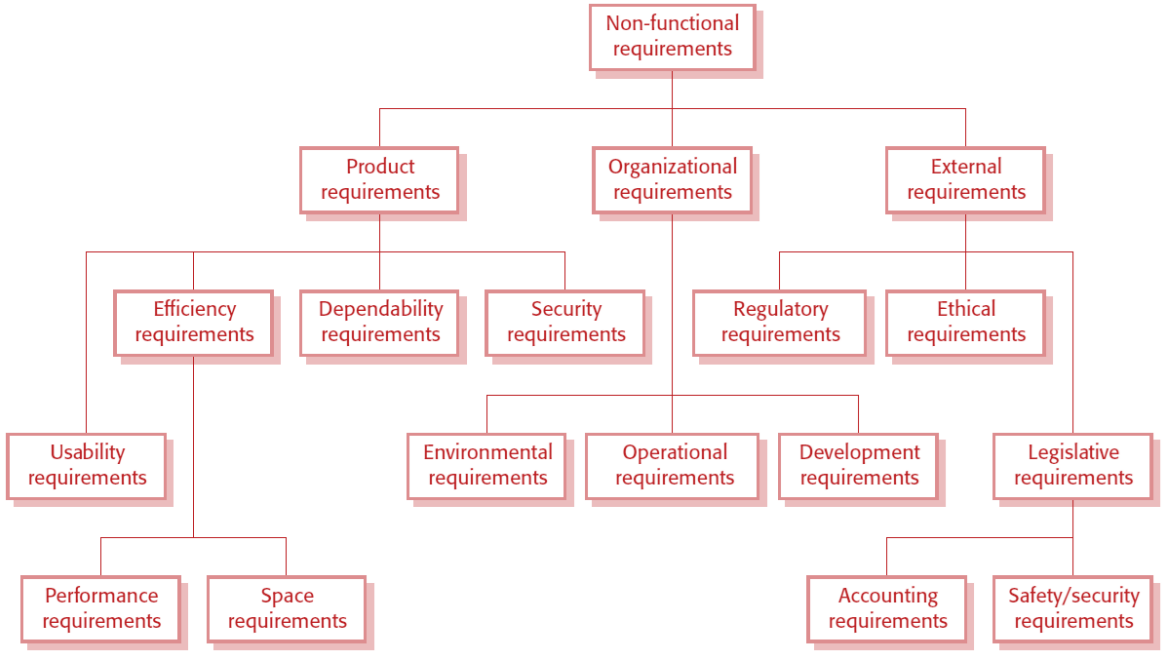
Types of requirement

- **Functional requirements**
  - Statements about the services the system **should provide**
    - How the system should react to particular inputs, and how the system should behave in particular situations
  - Sometimes it also explicitly state what the system **should not do**

Requirements Engineering

Types of requirement

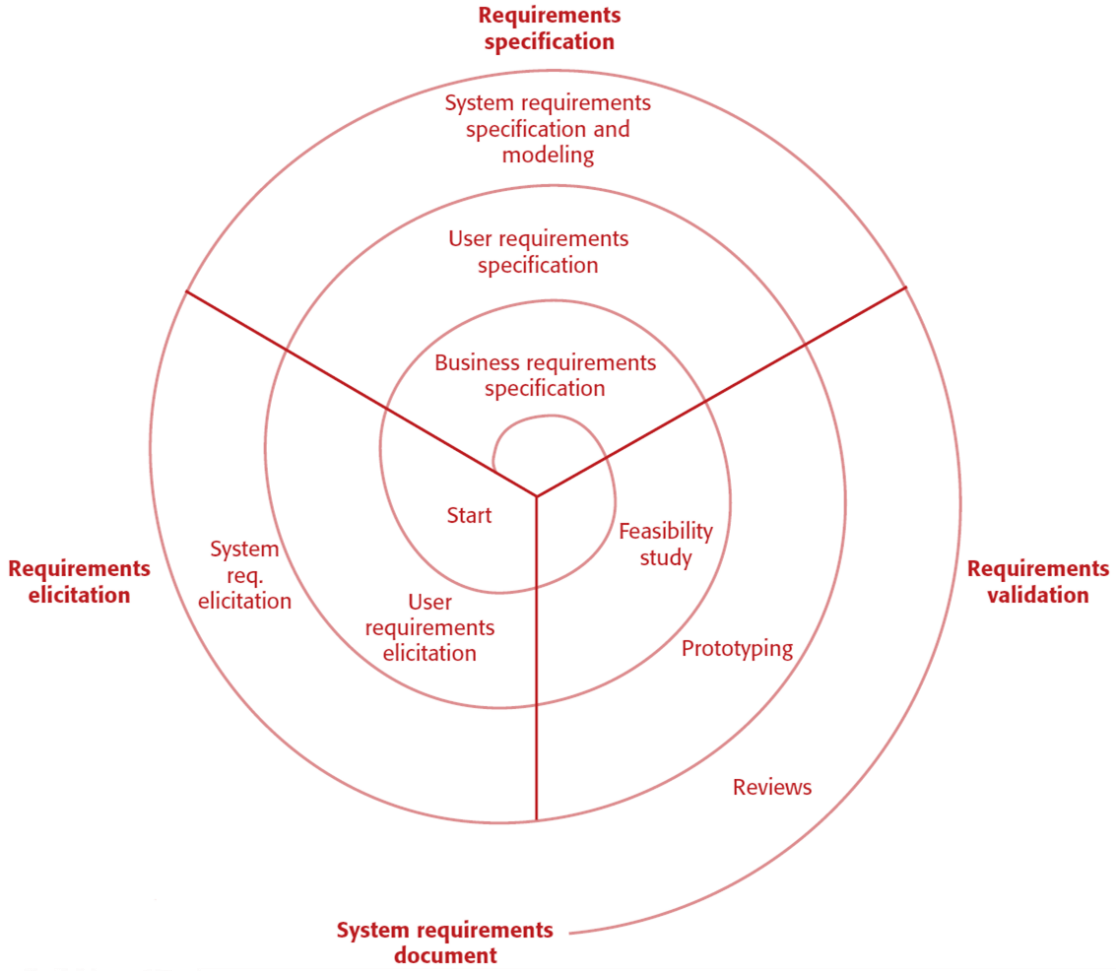
- **Non-functional requirements**
  - These are constraints on the services or functions offered by the system
    - Include timing constraints, constraints on the development process, and constraints imposed by standards
    - Often apply to the global system not to individual features or services



Requirements Engineering

About the process

- **Key activities**
  - **Elicitation and Analysis.** Discovering requirements by interacting with stakeholders
  - **Specification.** Converting these requirements into a standard form
  - **Validation.** Checking that the requirements actually define the system that the customer wants

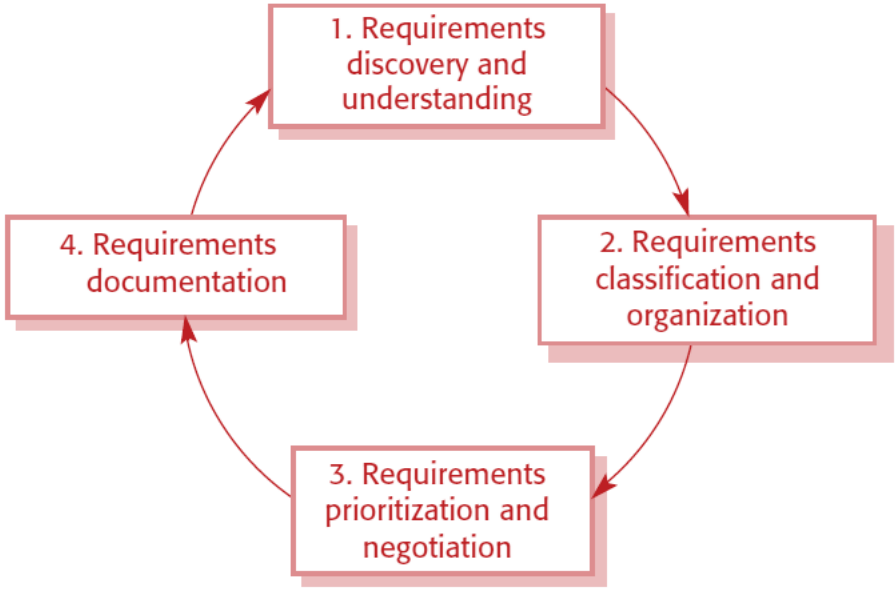


Requirements Engineering

About the process

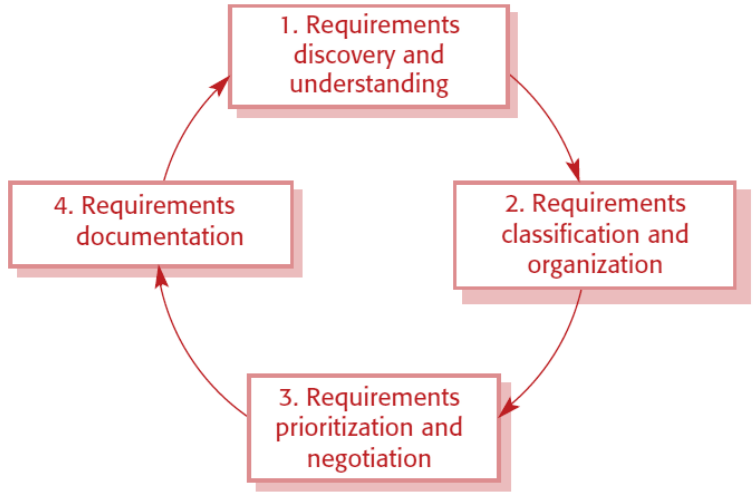
- **Key activities**

1. Interacting with stakeholders of the system to discover their requirements
2. Takes the unstructured collection of requirements, groups related requirements and organizes them into coherent clusters
3. Concerned with prioritizing requirements and finding and resolving conflicts
4. Requirements are documented and input into the next round of the spiral



Requirements Engineering

Elicitation and Analysis



- **Techniques**

1. **Interviewing.**
   - **Open interviews** where there is no agenda
   - **Closed interviews** when the stakeholders answers a predefined set of questions

2. **Ethnography.**
   - **Observational technique** that can be used to understand operational processes and help derive requirements for software to support them
   - **An analyst immerses** in the working environment

Requirements Engineering

Specification

- **Definition**

  Process of writing down the user and system requirements in a requirements document

  Requirements Document

  1. The official statement of what is required of the system developers
  2. Should include both a definition of **user requirements** and a **specification of the system requirements**
  3. It is **NOT** a design document. As far as possible, should set **WHAT** the system should do rather than **HOW** it should do it

Requirements Engineering

Specification

- **Natural language specification (two requirements)**
  - This method can be very vague. It depends on the writer ability on putting requirements into words
  - **It has little formalism**

> 3.2 The system shall measure the blood sugar and deliver insulin, if required, every 10 minutes. (*Changes in blood sugar are relatively slow, so more frequent measurement is unnecessary; less frequent measurement could lead to unnecessarily high sugar levels.*)
>
> 3.6 The system shall run a self-test routine every minute with the conditions to be tested and the associated actions defined in Table 1. (*A self-test routine can discover hardware and software problems and alert the user to the fact the normal operation may be impossible.*)

Requirements Engineering

Specification

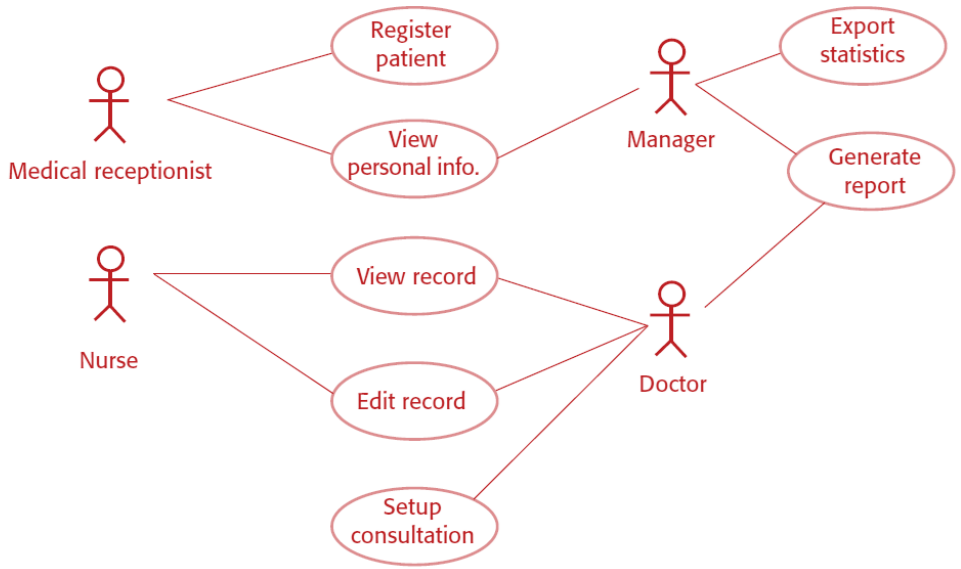
- **Alternatives to Natural Language specification**
  - These methods can sometimes be combined to illustrate deeper some customer requirements
  - **It is more work intensive**

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.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system. UML (unified modeling language) 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 they are reluctant to accept it as a system contract. (I discuss this approach, in Chapter 10, which covers system dependability.)

Requirements Engineering

Specification

- **Use Cases**
  - Use cases are a way of describing interactions between users and a system using a graphical model and structured text
  - **It is relatively easy to accomplish**



Requirements Engineering

Specification

- **Structured specification (one requirement)**
  - This method is **more detailed, thus, more precise** regarding customer requirements.
  - **It is more work intensive**

Insulin Pump/Control Software/SRS/3.3.2

Function	Compute insulin dose: Safe sugar level.
Description	Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.
Inputs	Current sugar reading (r2), the previous two readings (r0 and r1).
Source	Current sugar reading from sensor. Other readings from memory.
Outputs	CompDose—the dose in insulin to be delivered.
Destination	Main control loop.
Action:	CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered. (see Figure 4.14)
Requires	Two previous readings so that the rate of change of sugar level can be computed.
Precondition	The insulin reservoir contains at least the maximum allowed single dose of insulin.
Postcondition	r0 is replaced by r1 then r1 is replaced by r2.
Side effects	None.

Requirements Engineering

Specification

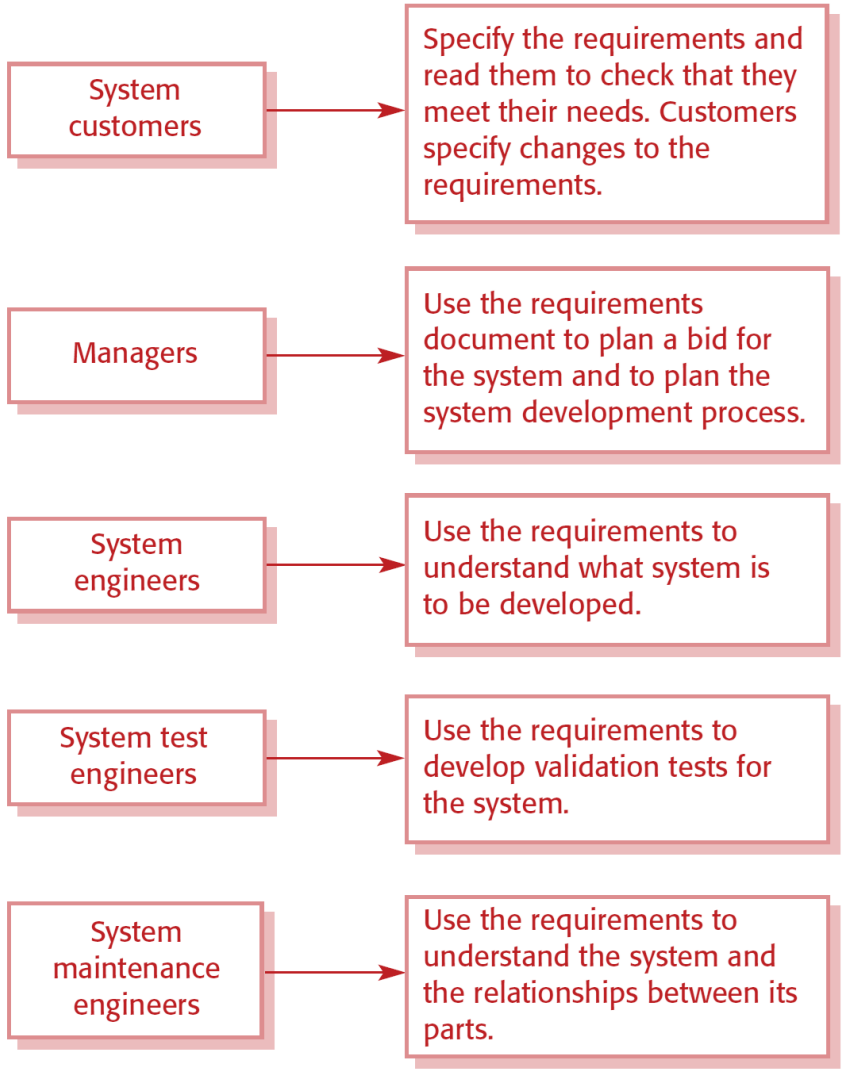
- **Software Requirements document**
  - Depends on the type of software being developed
  - For a complex engineering system that includes hardware and software developed by different companies the requirements document is likely to be long and detailed
  - For an in-house software product will leave out many of detailed chapters. The focus will be on defining the user requirements and high-level, nonfunctional system requirements

Chapter	Description
Preface	This defines 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 describes 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 defines 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 nonfunctional 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 presents 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 describes the functional and nonfunctional requirements in more detail. If necessary, further detail may also be added to the nonfunctional requirements. Interfaces to other systems may be defined.
System models	This chapter includes graphical system models showing the relationships between the system components and the system and its environment. Examples of possible models are object models, data-flow models, or semantic data models.
System evolution	This describes 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 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 Engineering

Specification

- **Requirements document Users**
  - Requirements documents are essential when systems are outsourced for development, when different teams develop different parts of the system, and when a detailed analysis of the requirements is mandatory
  - The level of detail that you should include in a requirements document depends on the type of system that is being developed and the development process used



Requirements Engineering

Requirements Validation

- **Goals**
  1. Concerned with demonstrating that the requirements define the system that the customer really wants
  2. Requirements error costs are high, so validation is very important
     - Fixing a requirements error after delivery may **cost up to 100 times** the cost of fixing an implementation error

Requirements Engineering

Requirements Validation

- **Requirements Checking**

1. **Validity.** Does the system provide the functions that support the customer's needs ?
2. **Consistency.** Are there any requirements conflicts ?
3. **Completeness.** Are all functions required by the customer included ?
4. **Realism.** Can the requirements be implemented given available budget, timeframe and technology ?
5. **Verifiability.** Can the requirements be checked/tested ?

- **Validation Techiques**

1. **Requirements reviews**
   - **Systematic, regular manual reviews/analysis.** Executed while the requirements definition is being formulated, involving the client. Reviews may be formal (with documents) or informal
   - **Review checks.** Verifiability (is the requirement testable ?), Comprehensibility (is it understood ?), Traceability (is its origin clearly stated ?), Adaptability (can it be changed without impact on other requirements ?)

2. **Prototyping**
   - Using an executable model of the system to check requirements

3. **Test-case generation**
   - Developing tests for requirements to check testability

Requirements Engineering

Requirements Validation

- **Requirements Management**

The process of managing changing requirements during the requirements engineering process and system development

1. **Business/technical environment.** It may change during development
2. **Priority of requirements.** May change during the development process
3. **Some requirements may become obsolete**