Requirements Engineering [4]

- What are requirements?
- How to produce a requirements specification.

Classification I

- User Requirements
  - abstract
  - natural language + simple diagrams
  - what users want
- System Requirements
  - more concrete, detailed
  - natural + formal language + diagrams
  - what system provides
  - used as contract or product description

Requirements are about WHAT, not HOW.

Figure 4.1 User and system requirements

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. 10mg, 20 mg, etc.) 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.

A restaurant example

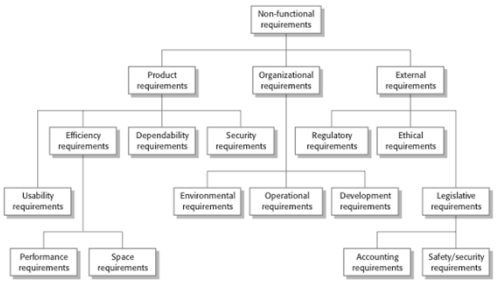
- User Requirements
  - food should look good
  - taste good
  - enough
  - fish
- System Requirements
  - chicken soup
  - baked salmon with …
  - chocolate mousse

”contract”

Classification II

- Functional Requirements
  what the system should do
- Non-functional Requirements
  constraints on the system
- Product
- Process
  constraints on the *development process*

Figure 4.3 Types of nonfunctional requirement



Problems and solutions

- Unclear terminology
- Example:
  a life insurance policy is *closed*

- Glossary
  dictionary of all ”technical” terms

Sales:
If it’s sold and paid for
Accounting:
When it has been paid back

Problems and solutions

- Unclear terminology
- Vague, untestable requirements
- Amalgamation, lack of organisation

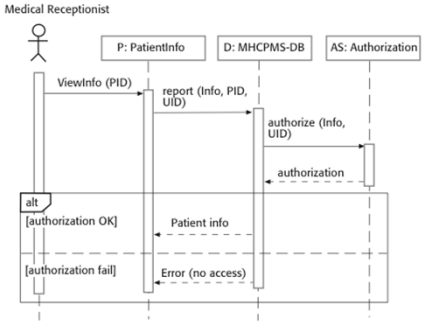
- Glossary
- A test scenario must be included
- Numbered items
- Standard formats 4.3.2
- Traceability

Problems and solutions

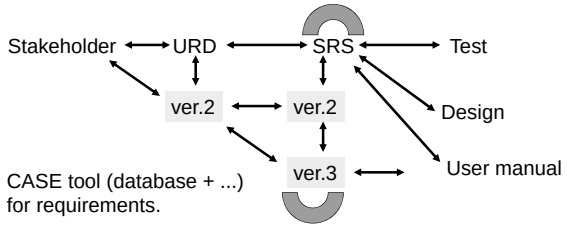
- Things that are not required occur
- Incomplete

- Rationale must be included
- Traceable to stakeholder
- Mandatory/Desired
- Formal model
- User manual
- Use cases

Figure 5.6 Sequence diagram for View patient information



Traceability



CASE = Computer Aided Software Engineering

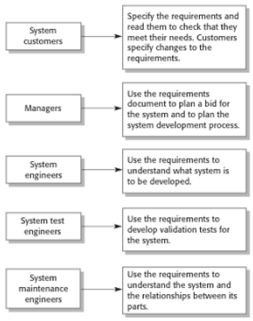
What’s in the requirements document

and who should read it?

Figure 4.7 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 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 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 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 might include 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 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.

Figure 4.6 Users of a requirements document



How do we get the requirements right?

- Interview with end-user
- use cases
- paper prototype
- prototype
- comparison with similar systems
- user manual

validation

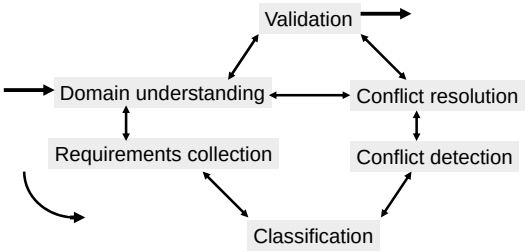
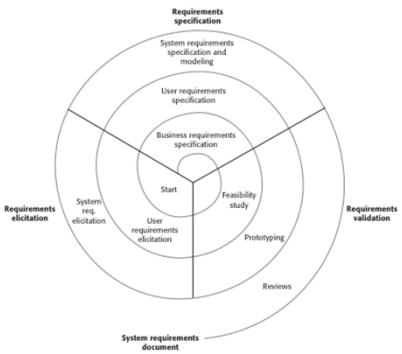


Figure 4.12 A spiral view of the requirements engineering process



Requirements checklist

- Understandable (properly explained)
- Validity (is this required, rationale)
- Verifiable (test cases included)
- Realism (feasibility, prototype)
- Complete (all cases covered, exceptions)
- Consistent
- Traceability

verification