Chapter 4 – Requirements Engineering

Ian Sommerville,

Software Engineering, 10th Edition

Pearson Education, Addison-Wesley

Note: These are a slightly modified version of Chapter 4 slides available from the author's site
http://iansommerville.com/software-engineering-book/

Topics covered

- ◇ Functional and non-functional requirements
- ◇ Requirements engineering processes
- ◇ Requirements elicitation
- ◇ Requirements specification
- ◇ Requirements validation
- ◇ Requirements change

Requirements engineering

- The process of establishing the services that a customer requires from a system and the constraints under which it operates and is developed.
- The system requirements are the descriptions of the system services and constraints that are generated during the requirements engineering process.
- Translation – Figure out what a system needs to do, write it down.

What is a **requirement**?

- Some function or characteristic that must exist in a project
- It may range from a high-level abstract statement to an extremely detailed description of a function

Types of requirement

◇ User requirements

- Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.
  - High level, easy to understand, potentially verbose

◇ System requirements

- A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.
  - Lower level, more complex, quite detailed

User and system requirements

User requirements definition

1. The Mentcare system 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 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, 20mg, etc) separate reports shall be created for each dose unit.
1.5 Access to drug cost reports shall be restricted to authorized users as listed on a management access control list.

Agile methods and requirements

- Many agile methods argue that producing detailed system requirements is a waste of time. Requirements change quickly, thus the requirements document is always out of date.
- This is problematic for systems that require pre-delivery analysis or systems developed by several teams.

Functional and non-functional requirements

Functional and non-functional requirements

◇ Functional requirements

- Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations (map directly to some code execution parts)
  - What the user sees, how they interact with the system, what the system does
- May state what the system should not do

◇ Non-functional requirements

- Constraints on (or characteristics of) the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
  - Behind the scenes, inform functional requirements
- Often apply to the system as a whole rather than individual features or services

Functional requirements

- Describe functionality or system services
- Dependent on the type of software, expected users and the type of system where the software is used
- Functional user requirements may be high-level statements of what the system should do
- Functional system requirements should describe the system services in detail

Requirements imprecision

- Problems arise when functional requirements are not precisely stated
- Ambiguous requirements may be interpreted in different ways by developers and users

Requirements completeness and consistency

- In principle, requirements should be both complete and consistent
- Complete
  - They should include descriptions of all functions required
- Consistent
  - There should be no conflicts or contradictions in the descriptions of the system facilities

Requirements completeness and consistency

- At what point does working towards completeness take away from the project?

"Good news! He said he only needs a few more weeks to finish the first draft of the Requirements Document."

Non-functional requirements

- These define system properties and constraints e.g., reliability, response time, storage requirements, platform
- Process requirements may also be specified mandating a particular IDE, programming language or development method
- While functional requirements map directly to running code, non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless.
- They may also force functional requirements. For instance a mobile application will require a touch based UI.

Non-functional requirements implementation

- 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

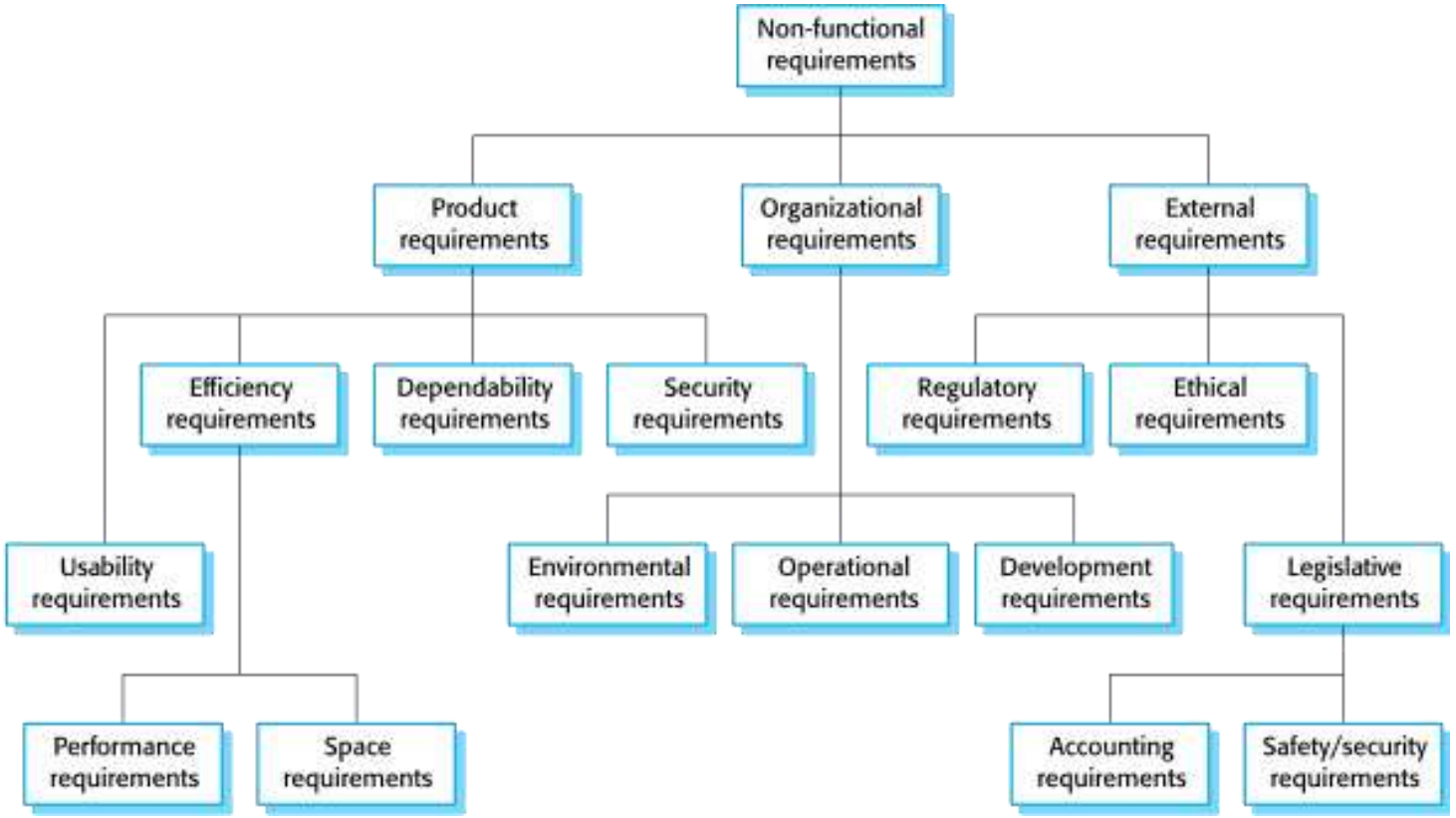
Non-functional classifications

◇ Product requirements
- Requirements which specify that the delivered product must behave in a particular way, e.g. execution speed, or reliability

◇ Organizational requirements
- Requirements which are a consequence of organizational policies and procedures, e.g. process standards used or implementation requirements

◇ External requirements
- Requirements which arise from factors which are external to the system and its development process, e.g. interoperability requirements and legislative requirements

Non-functional classifications



Examples of nonfunctional requirements in the Mentcare system

Product requirement
The Mentcare system shall be available to all clinics during normal working hours (Mon–Fri, 0830–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.

Organizational requirement
Users of the Mentcare system shall authenticate themselves using their health authority identity card. (Some nonfunctional requirements inform the need for specific functional requirements)

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

Mentcare system: functional requirements

- A user shall be able to search the appointments lists for all clinics.
- The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.
- Each staff member using the system shall be uniquely identified by his or her 8-digit employee number.

Metrics for specifying non-functional 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

Requirements elicitation

Requirements elicitation and analysis

- Sometimes called requirements elicitation or requirements discovery
- Involves technical staff working with customers to find out about the application domain, the services that the system should provide and the system's operational constraints.
- May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, and other. These are called *stakeholders.*
- Find requirements!

Requirements elicitation

- 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.
- Stages include:
  - Requirements discovery
  - Requirements classification and organization
  - Requirements prioritization and negotiation
  - Requirements specification

Problems of requirements elicitation

- Stakeholders don't know what they really want
- Stakeholders express requirements in their own terms
- Different stakeholders may have conflicting requirements
- Organizational and political factors may influence the system requirements
- The requirements change during the analysis process. New stakeholders may emerge and the business environment may change.

Problems of requirements elicitation

- ◇ Reality versus Perception
  - Customer may not understand why a computer can't do what a child can

Requirements discovery - Interviewing

- Formal or informal interviews with stakeholders are part of most RE processes
- Types of interview
  - Closed interviews based on pre-determined list of questions
  - Open interviews where various issues are explored with stakeholders
- Effective interviewing
  - Be open-minded, avoid pre-conceived ideas about the requirements and be willing to listen to stakeholders
  - Prompt the interviewee to get discussions going using a springboard question, a requirements proposal, or by working together on a prototype system

Problems with interviews

- Application specialists may use language to describe their work that isn't easy for the requirements engineer to understand.
  - Every profession has its own language
- Interviews are not good for understanding domain requirements
  - Requirements engineers cannot understand specific domain terminology (see above)
  - Some domain knowledge is so familiar that people find it hard to articulate or think that it isn't worth articulating
  - The odd case that is unlikely to come up in an interview

Requirements discovery - Ethnography

- A social scientist spends a considerable time observing and analyzing how people actually work
- People do not have to explain or articulate their work
- Social and organizational factors of importance may be observed
- Ethnographic studies have shown that work is usually richer and more complex than suggested by simple system models
- Time consuming

Stories and scenarios

- Scenarios and user stories are real-life examples of how a system can be used
- Stories and scenarios are a description of how a system may be used for a particular task
- Because they are based on a practical situations, stakeholders can relate to them and can comment on their situation with respect to the story.

Scenarios

- A structured form of user story
- Scenarios should include
  - A description of the starting situation
  - A description of the normal flow of events
  - A description of what can go wrong
  - Information about other concurrent activities
  - A description of the state when the scenario finishes

Requirements specification

Requirements specification

- The process of writing down the user and system requirements in a requirements document
- User requirements must be understandable by end-users and customers who do not have a technical background
- System requirements are more detailed requirements and may include more technical information
- The requirements may be part of a contract for the system development
  - It is therefore important that these are as complete as possible

Ways of writing a system requirements specification

Notation	Description
Natural language	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. (almost pseudocode)
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 and design

- In principle, requirements should state **what** the system should do and the design should describe **how** it does this
- In practice, requirements and design are inseparable
  - A system architecture may be designed to structure the requirements
  - The system may inter-operate with other systems that generate design requirements
  - The use of a specific architecture to satisfy non-functional requirements may be a domain requirement
  - This may be the consequence of a regulatory requirement

Natural language specification

- Requirements are written as natural language sentences supplemented by diagrams and tables
- Used for writing requirements because it is expressive, intuitive and universal. This means that the requirements can be understood by users and customers.

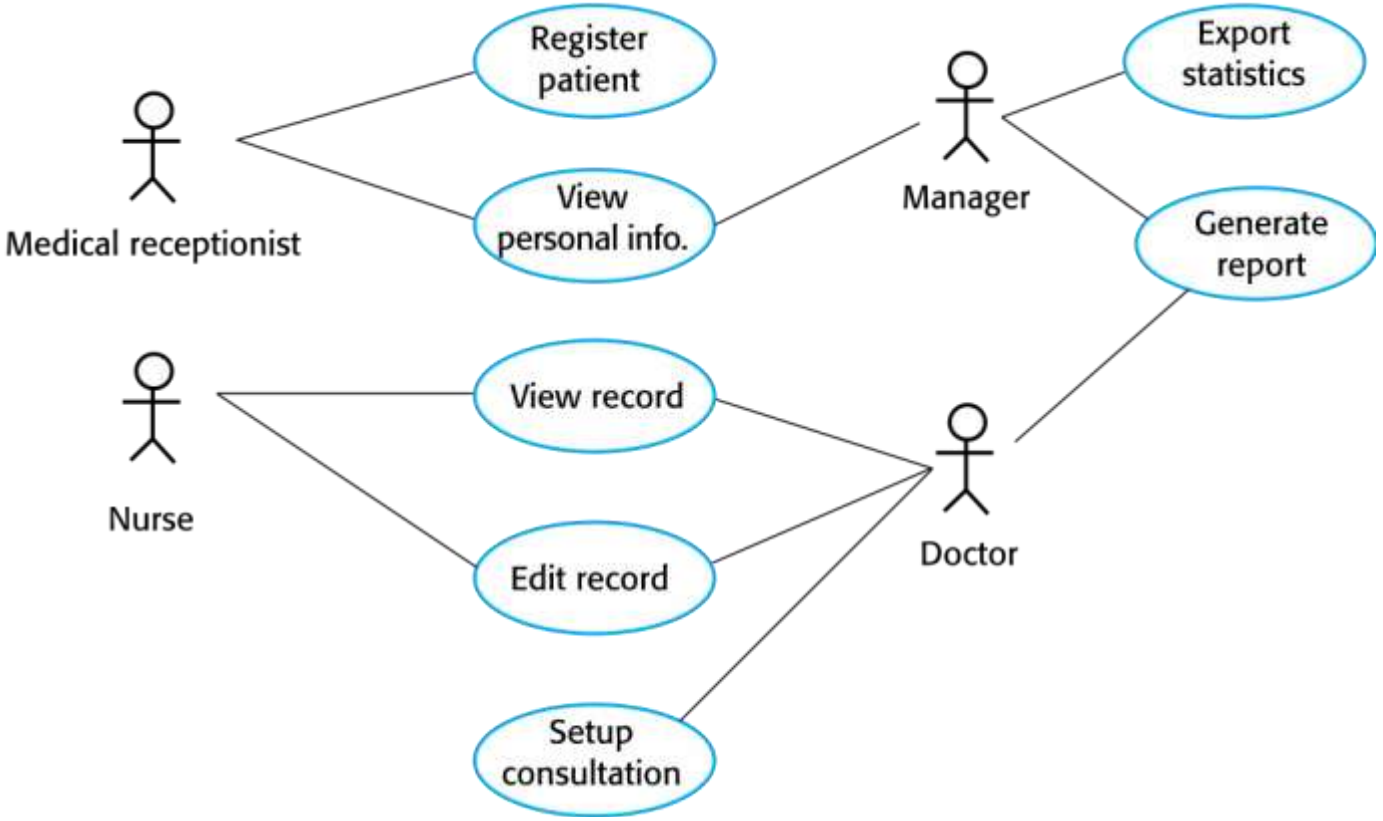
Problems with natural language

- Lack of clarity
  - Precision is difficult without making the document difficult to read
- Requirements confusion
  - Functional and non-functional requirements tend to be mixed-up
- Requirements amalgamation
  - Several different requirements may be expressed together

Use cases

- Use-cases are a kind of scenarios included in UML
- Use cases identify the actors in an interaction and which describe the interaction itself
- A set of use cases should describe all possible interactions with the system
- High-level graphical models supplemented by more detailed tabular description (see Chapter 5)
- UML sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system

Use cases for the Mentcare system



The software requirements document

- The software requirements document is the official statement of what is required of the system developers
- Should include both a definition of user requirements and a specification of the system requirements
- It is NOT a design document. As far as possible, it should set of WHAT the system should do rather than HOW it should do it.
- See general structure of a "req spec" document next, *but note that we use a "streamlined" version in Project Part #2*

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.

The structure of a requirements document

Chapter	Description
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.

Requirements validation

Requirements validation

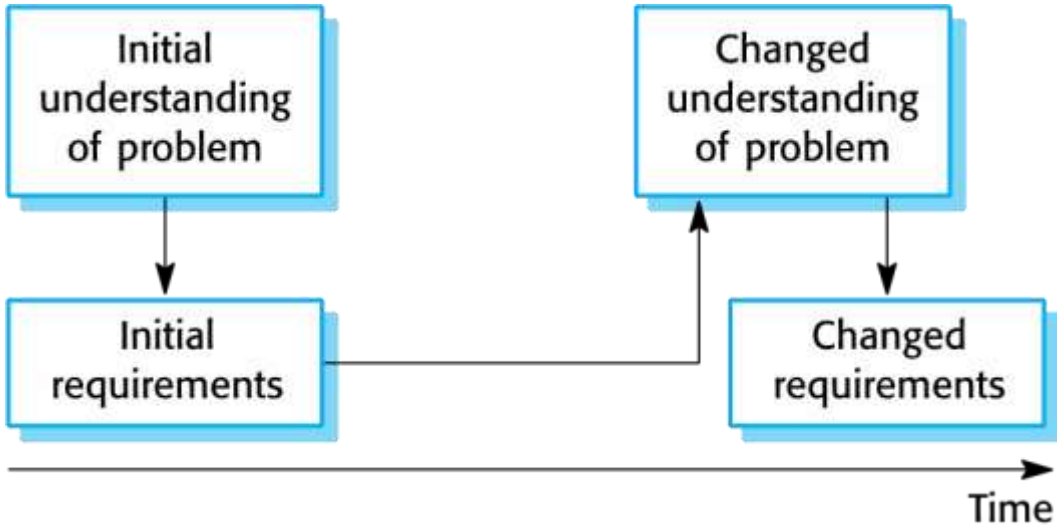
- Concerned with demonstrating that the requirements define the system that the customer really wants
- Requirements error costs are high so validation is very important
  - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error

Requirements checking

- 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 change

Requirements evolution



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
- You need to keep track of individual requirements and maintain links between dependent requirements so that you can assess the impact of requirements changes. You need to establish a formal process for making change proposals and linking these to system requirements.

Key points

- Requirements for a software system set out what the system should do and define constraints on its operation and implementation
- Functional requirements are statements of the services that the system must provide or are descriptions of how some computations must be carried out
- Non-functional requirements often constrain the system being developed and the development process being used
  - They often relate to the emergent properties of the system and therefore apply to the system as a whole

Key points

- The requirements engineering process is an iterative process that includes requirements elicitation, specification, and validation
- You can use a range of techniques for requirements elicitation including interviews and ethnography. User stories and scenarios may be used to facilitate discussions.
- Requirements specification is the process of formally documenting the user and system requirements and creating a software requirements document

Key points

- The software requirements document is an agreed statement of the system requirements. It should be organized so that both system customers and software developers can use it.
- Requirements validation is the process of checking the requirements for validity, consistency, completeness, realism and verifiability
- Business, organizational and technical changes inevitably lead to changes to the requirements for a software system. Requirements management is the process of managing and controlling these changes.