Requirements Engineering

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.

Requirements abstraction (Davis)

“If a company wishes to let a contract for a large software development project, it must define its needs in a sufficiently abstract way that a solution is not pre-defined. The requirements must be written so that several contractors can bid for the contract, offering, perhaps, different ways of meeting the client organization’s needs. Once a contract has been awarded, the contractor must write a system definition for the client in more detail so that the client understands and can validate what the software will do. **Both of these documents may be called the requirements document for the system**.”

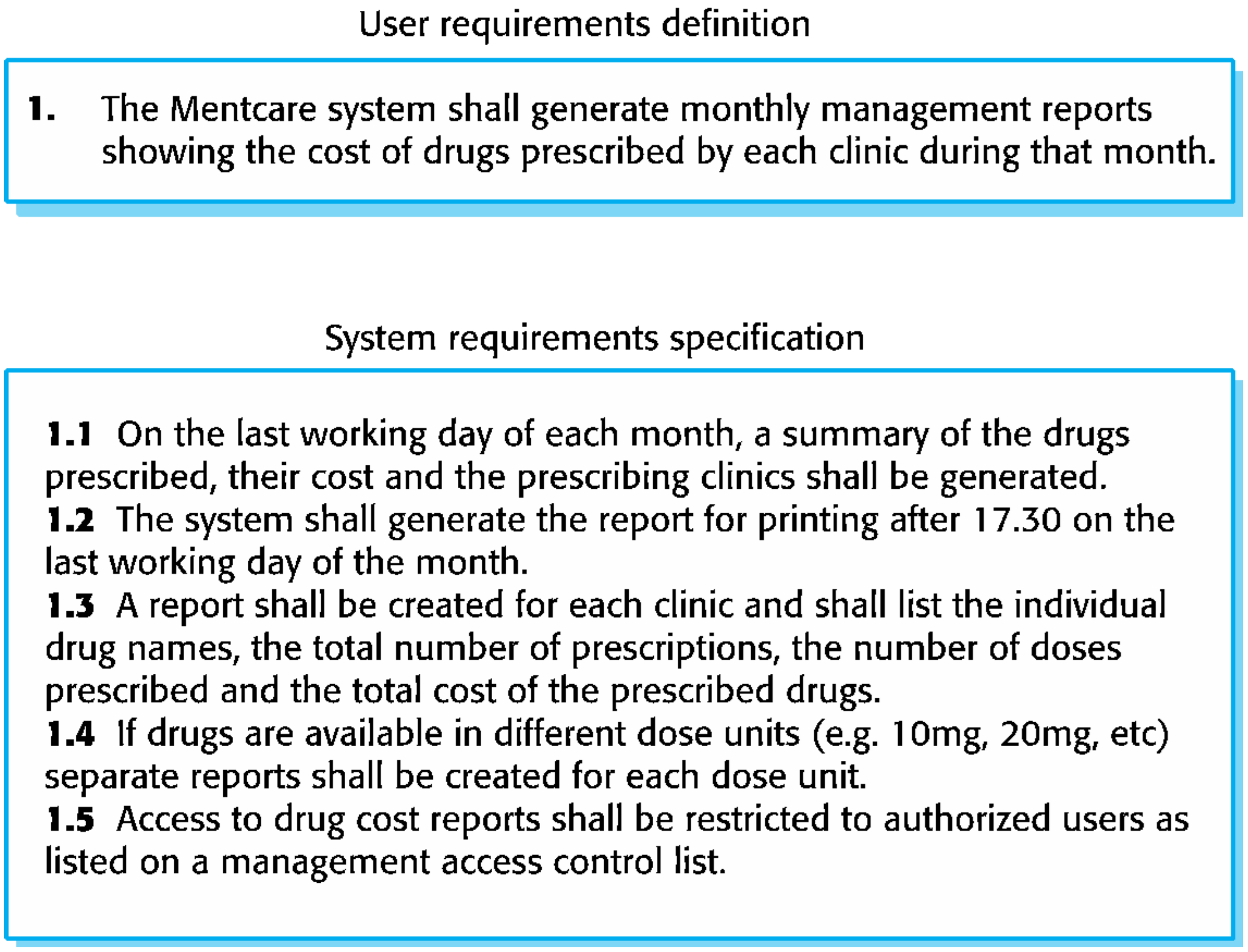
What is a requirement?

- ✓ It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.
- ✓ This is inevitable as requirements may serve a dual function
  - ✓ May be the basis for a bid for a contract - therefore must be open to interpretation;
  - ✓ May be the basis for the contract itself - therefore must be defined in detail;
  - ✓ Both these statements may be called requirements.

Types of requirement

- User requirements
  - Statements in natural language plus diagrams of the services the system provides and its operational constraints.
  - Written for customers.
- 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.
  - Required for developers.

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.

System stakeholders

- ✓ Any person or organization who is affected by the system in some way and so who has a legitimate interest
- ✓ Stakeholder types
  - End users
  - System managers and owners
  - System developers
  - External stakeholders like supplier and government making policies

Stakeholders in the Mentcare system

- ✓ Patients whose information is recorded in the system.
- ✓ Doctors who are responsible for assessing and treating patients.
- ✓ Nurses who coordinate the consultations with doctors and administer some treatments.
- ✓ Medical receptionists who manage patients' appointments.
- ✓ IT staff who are responsible for installing and maintaining the system.
- ✓ A medical ethics manager who must ensure that the system meets current ethical guidelines for patient care.
- ✓ Health care managers who obtain management information from the system.
- ✓ Medical records staff who are responsible for ensuring that system information can be maintained and preserved, and that record keeping procedures have been properly implemented.

Agile methods and requirements

- ✓ RE: Gather system requirements before taking go ahead decision of system development. This RE requires a high-level view of system requirement. Then perform feasibility study, which tries to assess whether or not the system is technically and financially feasible. It help in deciding whether or not to go ahead. The RE document may be part of the system development contract.
- ✓ Many agile methods argue that producing detailed system requirements is a waste of time as requirements change so quickly. The requirements document is therefore always out of date. Agile methods usually use incremental requirements engineering.
- ✓ Agile method is practical for business systems but problematic for systems that require pre-delivery analysis (e.g. critical systems) or systems developed by several teams.

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.
  - ▪ May state what the system should not do.
- ✓ Non-functional requirements
  - ▪ Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
  - ▪ Often apply to the system as a whole rather than individual features or services.
- ✓ The system requirements do not just specify the services or the features of the system that are required; they also specify the necessary functionality to ensure that these services/features are delivered effectively. Example security.

Functional requirements

- ✓ Describe functionality or system services.
- ✓ Depend 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. They should be written in natural language so that system users and managers can understand them.
- ✓ Functional system requirements should describe the system services in detail. They expand the user requirements and are written for system developers. They should describe the system functions, their inputs and outputs, and exceptions in detail.
- ✓ System requirements vary from general requirements covering what the system should do to very specific requirements reflecting local ways of working or an organization's existing systems.
- ✓ If an organization decides that an existing off-the-shelf system software product can meet its needs, then there is less effort on developing a detailed functional specification and more focused on how it is to be delivered and organized.

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.

Requirements imprecision

- ✓ Problems arise when functional requirements are not precisely stated.
- ✓ Ambiguous requirements may be interpreted in different ways by developers and users.
- ✓ The rationale for this requirement is that patients with mental health problems are sometimes confused. They may have an appointment at one clinic but actually go to a different clinic. If they have an appointment, they will be recorded as having attended, regardless of the clinic.
- ✓ Consider the term ‘search’ in requirement 1
  - ✓ User intention – search for a patient name across all appointments in all clinics;
  - ✓ Developer interpretation – search for a patient name in an individual clinic. User chooses clinic then search.

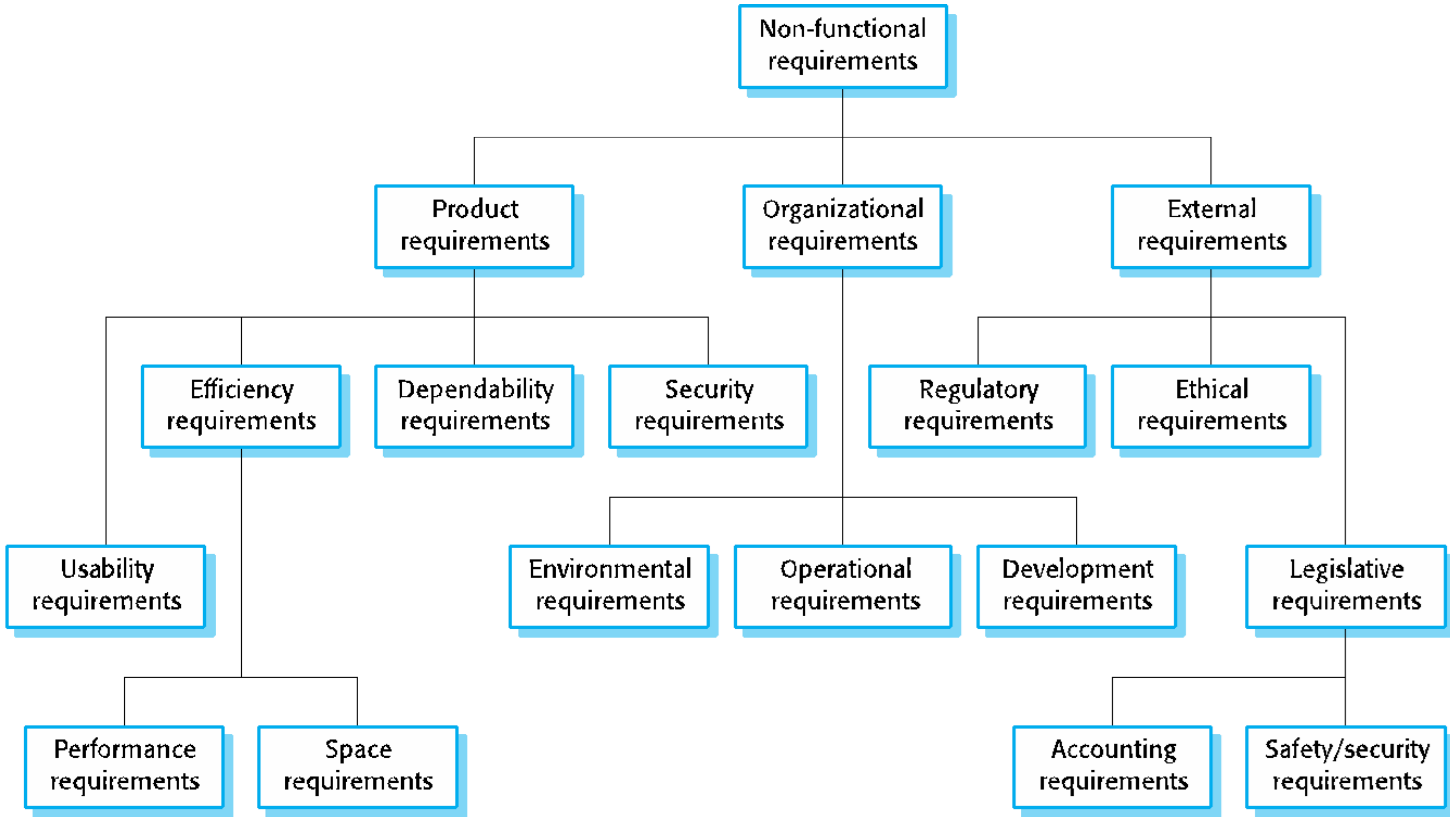
Requirements completeness and consistency

- ✓ In principle, requirements should be both complete and consistent. Complete means they should include descriptions of all facilities required. Consistent means there should be no conflicts or contradictions in the descriptions of the system facilities.
- ✓ In practice, it is impossible to achieve requirements consistency and completeness for large, complex systems because they are easy to make mistakes and omissions while writing specifications for large, complex systems and they have many stakeholders, with different backgrounds and expectations.
- ✓ Inconsistencies may not be obvious during the requirements specification but they may only be discovered after deeper analysis or during system development.

Non-functional requirements

- ✓ These define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, data representations used in interface with other systems.
- ✓ Process requirements may also be specified mandating a particular IDE, programming language or development method.
- ✓ Non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless. Example, if an aircraft system does not meet its reliability requirements, it will not be certified as safe for operation.

Types of nonfunctional requirement



Non-functional requirements implementation

It is easy to identify which system components implement specific functional requirements but this is often more difficult with non-functional requirements because:

- ✓ 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.
- ✓ A single non-functional requirement, such as a security requirement, may generate a number of related functional requirements that define system services that are required.
  - It may also generate requirements that restrict existing requirements.

Non-functional classifications

Product requirements

- ✓ Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.

Organisational requirements

- ✓ Requirements which are a consequence of organisational policies and procedures e.g. development process requirements that specify programming language; and environmental requirements that specify the operating environment of the system. etc.

External requirements

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

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.

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

Privacy is very important issue in health care systems, and the requirement specifies that the system should be developed in accordance with a national privacy standard.

Metrics for specifying 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

Try to write non-functional requirements quantitatively using these metrics so that they can be objectively tested. Problems here:

- Customers find it difficult to translate their goals into measurable requirements, example there is no metric for maintainability.
- Customers may not be able to relate their needs to these specifications. Example, they don’t understand what some number defining the reliability (for example) means in terms of their everyday experience with computer systems.
- Customers neglect the importance of some non-functional requirements may be due to cost issues.

Requirements engineering processes

Requirements engineering processes

The processes used for RE vary widely depending on the application domain, the people involved and the organisation developing the requirements.

However, there are a number of generic activities common to all processes

- ✓ Requirements elicitation;
- ✓ Requirements analysis;
- ✓ Requirements validation;
- ✓ Requirements management.

In practice, RE is an iterative activity in which these processes are interleaved.

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, etc. These are called *stakeholders.*
- ✓ 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.

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.
- ✓ The requirements change during the analysis process. New stakeholders may emerge and the business environment may change.

Process activities

Requirements discovery

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

Requirements classification and organisation

- ✓ Groups related requirements and organises them into coherent clusters.

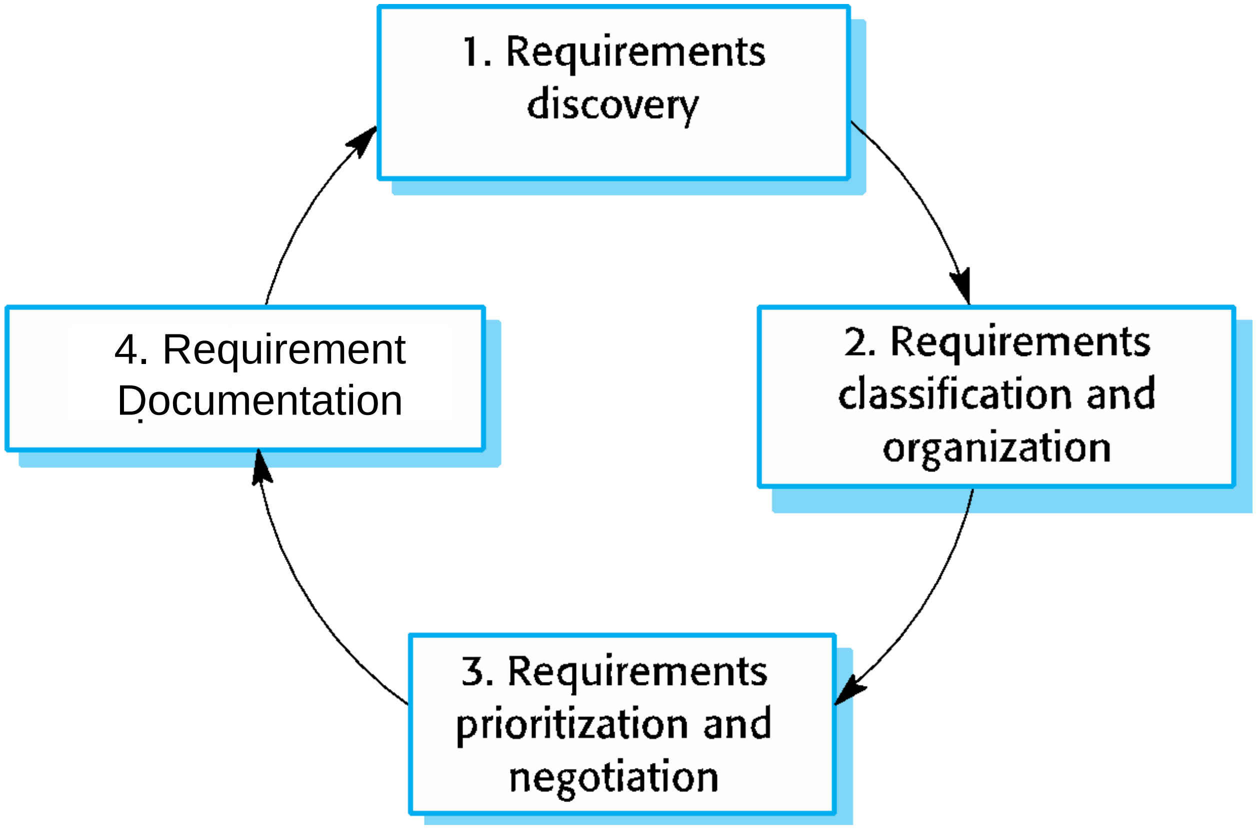
Prioritisation and negotiation

- ✓ Prioritising requirements and resolving requirements conflicts.

Requirements documentation

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

The requirements elicitation and analysis process


1. Requirements discovery
2. Requirements classification and organization
3. Requirements prioritization and negotiation
4. Requirement Documentation

- Requirements elicitation is an iterative process.
- 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 has been produced.

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 are willing to listen to stakeholders.
- ✓ Prompt the interviewee to get discussions going using questions or by working together on a prototype system.

Interviews in practice

- ✓ Normally a mix of closed and open-ended interviewing.
- ✓ Interviews are good for getting an overall understanding of what stakeholders do and how they might interact with the system.
- ✓ Interviewers need to be open-minded without pre-conceived ideas of what the system should do
- ✓ You need to prompt the use to talk about the system by suggesting requirements rather than simply asking them what they want.
- ✓ Interviews are good for getting an overall understanding of what stakeholders do, how they might interact with the new system, and the difficulties that they face with current systems.
- ✓ Unless you have a system prototype to demonstrate, you should not expect stakeholders to suggest specific and detailed requirements.

Problems with interviews

- ✓ Application specialists may use language to describe their work that isn’t easy for the requirements engineer to understand.
- ✓ Interviews are not good for understanding domain requirements
  - ▪ Requirements engineers cannot understand specific domain terminology;
  - ▪ Some domain knowledge is so familiar that people find it hard to articulate or think that it isn’t worth articulating. For example, for a librarian, it goes without saying that all acquisitions are catalogued before they are added to the library.

Ethnography

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

Scope of ethnography

- ✓ Requirements that are derived from the way that people actually work rather than the way I which process definitions suggest that they ought to work.
- ✓ Usefulness: In practice, people never follow formal processes. For example, air traffic controllers may switch off a conflict alert system that detects aircraft with intersecting flight paths, even though normal control procedures specify that it should be used.
- ✓ Requirements that are derived from cooperation and awareness of other people’s activities. Awareness of what other people are doing leads to changes in the ways in which we do things.
- ✓ For example, air traffic controllers (ATCs) may use an awareness of other controlles’ work to predict the number of aircraft that will be entering their control sector. They then modify their control strategies depending on that predicted workload.
- ✓ Ethnography is effective for understanding existing processes but cannot identify new features that should be added to a system.

Ethnography

- ✓ Ethnography can be combined with prototyping to reduce the prototype refinement cycles. Prototyping utilizes the ethnography by identifying problems and then look for the answers during the next phase of the system study.
- ✓ The problem with ethnography is that it studies existing practices which may have some historical basis which is no longer relevant.

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 situation, stakeholders can relate to them and can comment on their situation with respect to the story.
- ✓ Stories are written as narrative text and present a high-level description of system use; scenarios are usually structured with specific information collected such as inputs and outputs.

Scenarios

A scenario starts with an outline of the interaction. During the elicitation process, details are added to create a complete description of that interaction. Generally, a scenario may include:

- A description of what the system and users expect when the scenario starts.
- A description of the normal flow of events in the scenario.
- A description of what can go wrong and how resulting problems can be handled.
- Information about other activities that might be going on at the same time.
- A description of the system state when the scenario ends.

Requirements specification

Requirements specification

- ✓ The process of writing down the user and system requirements in a requirements document.
- ✓ User requirements have to 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.
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;

Guidelines for writing requirements

- ✓ Invent a standard format and use it for all requirements.
- ✓ Use language in a consistent way. Use shall for mandatory requirements, should for desirable requirements.
- ✓ Use text highlighting to identify key parts of the requirement.
- ✓ Avoid the use of computer jargon.
- ✓ Include an explanation (rationale) of why a requirement is necessary and who proposed it. It will be helpful in requirement change.

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

Will discuss later….

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.
- ✓ Requirements documents are essential when different teams develop different parts of the system, and when a detailed analysis of the requirements is mandatory.
- ✓ Agile methods argue that requirements change so rapidly that a requirements document is out of date as soon as it is written.
- ✓ For systems having unstable requirements, it is useful to write a short supporting document that defines system requirements. One can easily forget the requirements when focusing on the functional requirements for the next system release.
- ✓ 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.

Requirements document variability

- Information in requirements document depends on type of system and the approach to development used.
    - Critical systems need detailed requirements because safety and security have to be analyzed in detail to find possible requirements errors.
    - When the system is to be developed by a separate company (e.g., through outsourcing), the system specifications need to be detailed and precise.
    - If an in-house, iterative development process is used, the requirements document can be less detailed. Details can be added to the requirements and ambiguities resolved during development of the system.
- Systems developed incrementally will, typically, have less detail in the requirements document.
- Requirements documents standards have been designed e.g. IEEE standard. These are mostly applicable to the requirements for large systems engineering projects.

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? It means writing a set of tests that can demonstrate that the delivered system meets each specified requirement. It helps to reduce the potential dispute between customer and contractor.

Requirements validation techniques

- ✓ Requirements reviews
  - ✓ Systematic manual analysis of the requirements.
- ✓ Prototyping
  - ✓ Using an executable model of the system to check requirements.
- ✓ Test-case generation
  - ✓ Developing tests for requirements to check testability.
  - ✓ Creating tests for requirements in the validation process, will often reveals the difficulty in satisfying the requirements. If a test is difficult or impossible to design, this usually means that the requirements will be difficult to implement and should be reconsidered.

Requirements reviews

- ✓ Regular reviews should be held while the requirements definition is being formulated.
- ✓ Both client and contractor staff should be involved in reviews.
- ✓ Reviews may be formal (with completed documents) or informal. Good communications between developers, customers and users can resolve problems at an early stage.

Thanks