Elements of Software Engineering and Information Systems

Lecture 4 – Requirements Engineering

Prof. Ing. Lelio Campanile, PhD

Corso di Laurea Magistrale in Data Science
Dipartimento di Matematica e Fisica
Università degli Studi della Campania Luigi Vanvitelli

A.A. 2025/2026

Topics covered

Università degli Studi della Campania *Luigi Vanvitelli*

Definition

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.

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.

Università degli Studi della Campania *Luigi Vanvitelli*

Requirements abstraction (Davis)

Quote

"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."

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.

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.

Readers of different types of requirements specification

User requirements

- Client managers
- System end-users
- Client engineers
- Contractor managers
- System architects

System requirements

- System end-users
- Client engineers
- System architects
- Software developers

Università degli Studi della Campania *Luigi Vanvitelli*

System stakeholders

Definition

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
- System owners
- External stakeholders

Università degli Studi della Campania *Luigi Vanvitelli*

Stakeholders in the Mentcare system (1/2)

- **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.

Università degli Studi della Campania *Luigi Vanvitelli*

Stakeholders in the Mentcare system (2/2)

- 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

- 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** and may express requirements as *user stories*.
- This is practical for business systems but problematic for systems that require pre-delivery analysis (e.g. critical systems) or systems developed by several teams.

Università degli Studi della Campania *Luigi Vanvitelli*

Feasibility study

- Some understanding of the requirements may have to be developed before a decision is made to go ahead with the development of a system.
- This early-stage RE gives a high-level view of what the system might do and the benefits that it might provide.
- This may then be considered in a feasibility study that must answer:

Key Questions

- Does the system contribute to the overall objectives of the organization?
- Can the system be implemented within schedule and budget using current technology?
- Can the system be integrated with other systems that are used?

Università degli Studi della Campania *Luigi Vanvitelli*

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. 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.

Domain requirements

Constraints on the system from the domain of operation.

Università degli Studi della Campania *Luigi Vanvitelli*

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.
- Functional **system** requirements should describe the system services in detail.

Mentcare system: functional requirements

Examples

- 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.

Example: 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

They should include descriptions of all facilities required.

Consistent

There should be no conflicts or contradictions in the descriptions of the system facilities.

In practice

Because of system and environmental complexity, it is impossible to produce a complete and consistent requirements document.

Non-functional requirements

- These define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular IDE, programming language or development method.

Critical importance

Non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless.

Università degli Studi della Campania *Luigi Vanvitelli*

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.
- 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.

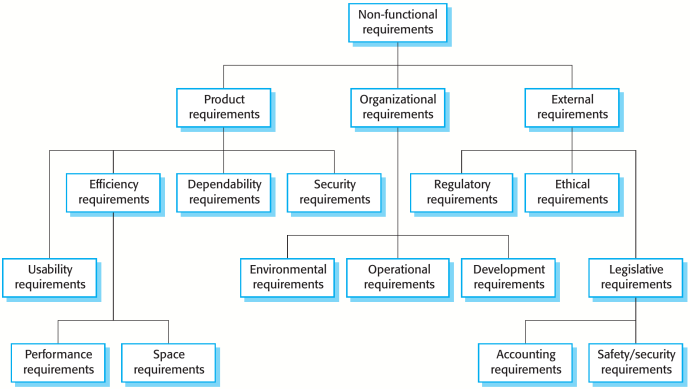
Organisational requirements

Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, 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.

Types of nonfunctional requirement



Examples of non-functional 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 GDPR.

Goals and requirements

- Non-functional requirements may be very difficult to state precisely and imprecise requirements may be difficult to verify.

Goal
: A general intention of the user such as ease of use.

Verifiable non-functional requirement
: A statement using some measure that can be objectively tested.

Goals are helpful to developers as they convey the intentions of the system users.

Goal example: Usability requirements

Goal

The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized.

Testable non-functional requirement

Medical staff shall be able to use all the system functions after four hours of training. After this training, the average number of errors made by experienced users shall not exceed two per hour of system use.

Università degli Studi della Campania *Luigi Vanvitelli*

Metrics for specifying nonfunctional requirements

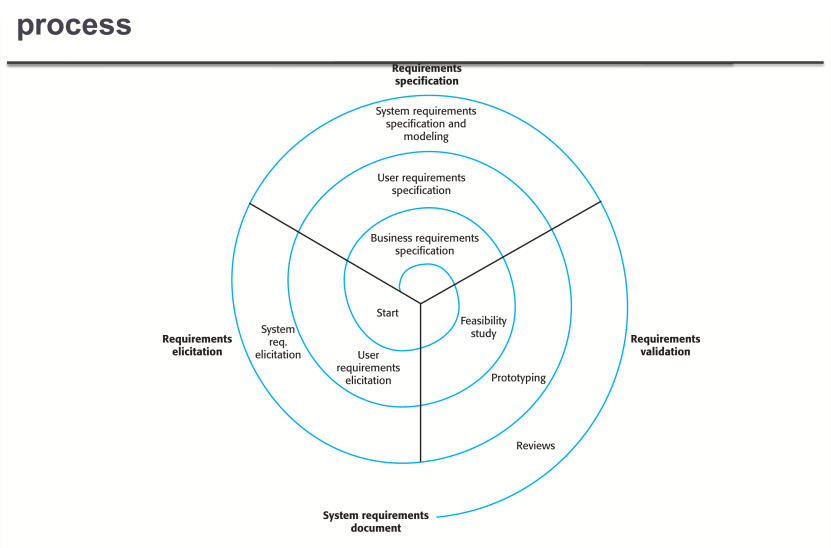
Property	Measure
Speed	Processed transactions/second; User/event response time; Screen refresh time
Size	Mbytes; Number of ROM chips
Ease of use	Training time; Number of help frames
Reliability	Mean time to failure; Probability of unavailability; Rate of failure occurrence; Availability
Robustness	Time 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

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.

A spiral view of the requirements engineering process



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**.

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, etc.

Stages

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

Università degli Studi della Campania *Luigi Vanvitelli*

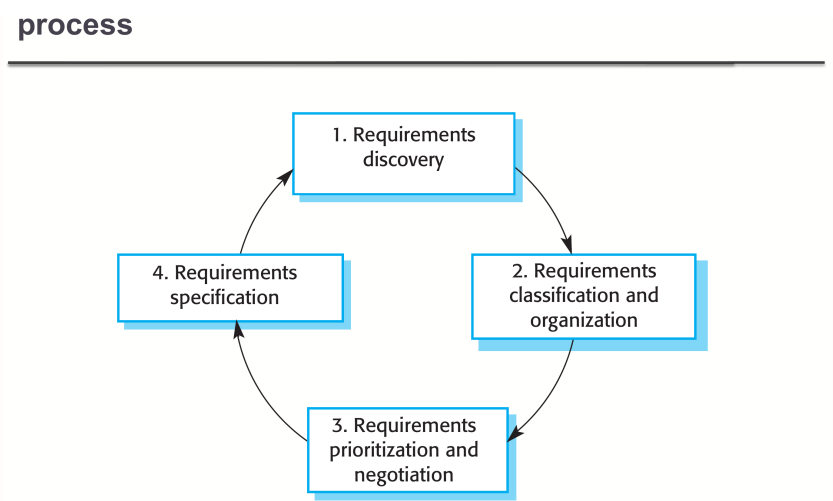
Problems of requirements elicitation

Common challenges

- Stakeholders don't know what they really want.
- Stakeholders express requirements in their own terms.
- Different stakeholders may have conflicting requirements.
- Organisational 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.

Università degli Studi della Campania *Luigi Vanvitelli*

The requirements elicitation and analysis process



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

Università degli Studi della Campania *Luigi Vanvitelli*

Requirements discovery

- The process of gathering information about the required and existing systems and distilling the user and system requirements from this information.
- Interaction is with system stakeholders from managers to external regulators.
- Systems normally have a range of stakeholders.

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
- Prompt the interviewee using a springboard question, a requirements proposal, or a prototype system

Università degli Studi della Campania *Luigi Vanvitelli*

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 user to talk about the system by suggesting requirements rather than simply asking them what they want.

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:

Domain knowledge issues

- 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.

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.

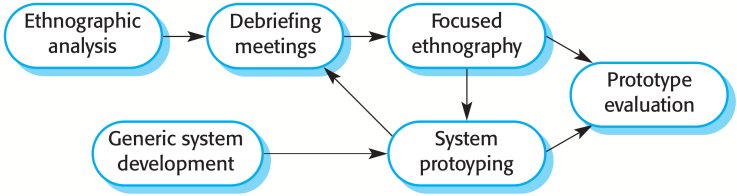
Scope of ethnography

- Requirements that are derived from the way that people **actually work** rather than the way in which process definitions suggest that they ought to work.
- 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.
- Ethnography is effective for understanding existing processes but cannot identify new features that should be added to a system.

Focused ethnography

- Developed in a project studying the air traffic control process.
- Combines ethnography with prototyping.
- Prototype development results in unanswered questions which focus the ethnographic analysis.
- The problem with ethnography is that it studies existing practices which may have some historical basis which is no longer relevant.

analysis


Ethnographic analysis
Debriefing meetings
Focused ethnography
Prototype evaluation
Generic system development
System protoyping

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.

Photo sharing in the classroom (iLearn)

Jack is a primary school teacher in Ullapool (a village in northern Scotland). He has decided that a class project should be focused around the fishing industry in the area, looking at the history, development and economic impact of fishing. As part of this, pupils are asked to gather and share reminiscences from relatives, use newspaper archives and collect old photographs related to fishing and fishing communities in the area.

Jack sends an email to a primary school teachers group to see if anyone can recommend an appropriate system. Two teachers reply and both suggest that he uses KidsTakePics, a photo sharing site that allows teachers to check and moderate content.

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.

Uploading photos – iLearn scenario (1/2)

Initial assumption

A user or a group of users have one or more digital photographs to be uploaded to the picture sharing site. These are saved on either a tablet or laptop computer. They have successfully logged on to KidsTakePics.

Normal flow

The user chooses upload photos and they are prompted to select the photos to be uploaded on their computer and to select the project name under which the photos will be stored. They should also be given the option of inputting keywords that should be associated with each uploaded photo. Uploaded photos are named by creating a conjunction of the user name with the filename of the photo on the local computer.
On completion of the upload, the system automatically sends an email to the project moderator asking them to check new content.

Uploading photos – iLearn scenario (2/2)

What can go wrong

- No moderator is associated with the selected project. An email is automatically generated to the school administrator asking them to nominate a project moderator.
- Photos with the same name have already been uploaded by the same user. The user should be asked if they wish to re-upload, rename, or cancel.

System state on completion

User is logged on. The selected photos have been uploaded and assigned a status 'awaiting moderation'. Photos are visible to the moderator and to the user who uploaded them.

Università degli Studi della Campania *Luigi Vanvitelli*

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:

Important

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	Uses a language like a programming language, but with more abstract features to specify the requirements by defining an operational model. Now rarely used.
Graphical notations	Graphical models supplemented by text annotations are used; UML use case and sequence diagrams are commonly used.
Mathematical specifications	Based on mathematical concepts such as finite-state machines or sets. Unambiguous but most customers don't understand formal specifications.

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:

Why they 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.

Università degli Studi della Campania *Luigi Vanvitelli*

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.

Guidelines for writing requirements

Best practices

- 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.

Università degli Studi della Campania *Luigi Vanvitelli*

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.

Example requirements for the insulin pump software system

Requirement 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.)*

Requirement 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 that normal operation may be impossible.)*

Structured specifications

- An approach to writing requirements where the freedom of the requirements writer is limited and requirements are written in a standard way.
- This works well for some types of requirements e.g. requirements for embedded control systems but is sometimes too rigid for writing business system requirements.

Form-based specifications

Elements of a form-based specification

- Definition of the function or entity.
- Description of inputs and where they come from.
- Description of outputs and where they go to.
- Information about the information needed for the computation and other entities used.
- Description of the action to be taken.
- Pre and post conditions (if appropriate).
- The side effects (if any) of the function.

A structured specification of a requirement for an insulin pump (1/2)

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.

A structured specification of a requirement for an insulin pump (2/2)

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.
Requirements
Two previous readings so that the rate of change of sugar level can be computed.
Pre-condition
The insulin reservoir contains at least the maximum allowed single dose of insulin.
Post-condition r0 is replaced by r1 then r1 is replaced by r2.
Side effects None.

Tabular specification

- Used to supplement natural language.
- Particularly useful when you have to define a number of possible alternative courses of action.
- For example, the insulin pump system bases its computations on the rate of change of blood sugar level and the tabular specification explains how to calculate the insulin requirement for different scenarios.

Tabular specification of computation for an insulin pump

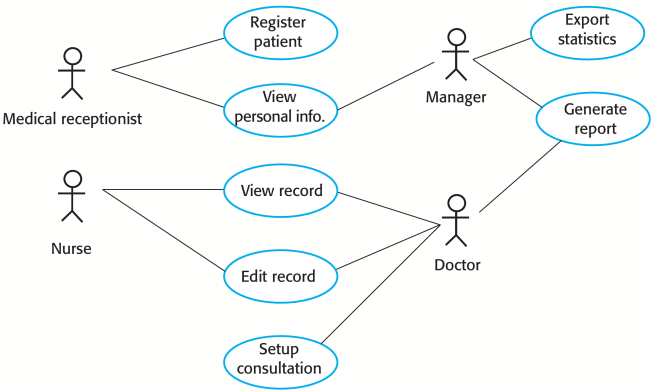
Condition	Action
Sugar level falling ($r2 < r1$)	CompDose = 0
Sugar level stable ($r2 = r1$)	CompDose = 0
Sugar level increasing and rate of increase decreasing $((r2 - r1) < (r1 - r0))$	CompDose = 0
Sugar level increasing and rate of increase stable or increasing $((r2 - r1) \geq (r1 - r0))$	CompDose = round$((r2 - r1)/4)$ If rounded result = 0 then CompDose = MinimumDose

Università degli Studi della Campania *Luigi Vanvitelli*

Use cases

- Use-cases are a kind of scenario that are included in the 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 model supplemented by more detailed tabular description.
- 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



Use cases for the Mentcare system: description

Setup Consultation use case

Setup consultation allows two or more doctors, working in different offices, to view the same patient record at the same time.
One doctor initiates the consultation by choosing the people involved from a dropdown menu of doctors who are online.
The patient record is then displayed on their screens, but only the initiating doctor can edit the record.
In addition, a text chat window is created to help coordinate actions.
It is assumed that a phone call for voice communication can be separately arranged.

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.

Important

It is NOT a design document. As far as possible, it should set out **WHAT** the system should do rather than **HOW** it should do it.

Users of a requirements document

User	Purpose
System customers	Specify the requirements and read them to check that they meet their needs. Specify changes to the requirements.
Managers	Use the requirements document to plan a bid for the system and to plan the system development process.
System engineers	Use the requirements to understand what system is to be developed.
System test engineers	Use the requirements to develop validation tests for the system.
System maintenance engineers	Use the requirements to understand the system and the relationships between its parts.

The structure of a requirements document

Chapter	Description
Preface	Define the expected readership and describe its version history.
Introduction	Describe the need for the system, its functions, and how it fits into overall business objectives.
Glossary	Define the technical terms used in the document.
User requirements definition	Describe the services provided for the user and nonfunctional system requirements.
System architecture	Present a high-level overview of the anticipated system architecture.
System requirements specification	Describe the functional and nonfunctional requirements in more detail.
System models	Include graphical system models showing relationships between components.
System evolution	Describe fundamental assumptions and any anticipated changes.
Appendices	Provide detailed, specific information related to the application.

Requirements document variability

- Information in requirements document depends on type of system and the approach to development used.
- 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.

Requirements Validation

Requirements validation

- Concerned with demonstrating that the requirements define the system that the customer really wants.

Cost of requirements errors

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?

Università degli Studi della Campania *Luigi Vanvitelli*

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.

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.

Review checks

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?

Requirements Change

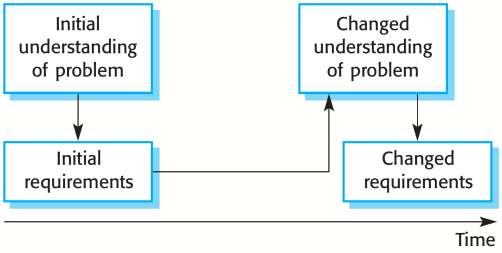
Changing requirements (1/2)

- The business and technical environment of the system always changes after installation.
  - New hardware may be introduced, it may be necessary to interface the system with other systems, business priorities may change, and new legislation and regulations may be introduced.
- The people who pay for a system and the users of that system are rarely the same people.
  - System customers impose requirements because of organizational and budgetary constraints. These may conflict with end-user requirements.

Changing requirements (2/2)

- Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory.
  - The final system requirements are inevitably a compromise between them, and, with experience, it is often discovered that the balance of support given to different users has to be changed.

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.

Requirements management planning

- Establishes the level of requirements management detail that is required.

Requirements management decisions

Requirements identification

Each requirement must be uniquely identified so that it can be cross-referenced with other requirements.

A change management process

The set of activities that assess the impact and cost of changes.

Traceability policies

Define the relationships between each requirement and between the requirements and the system design.

Tool support

Tools may range from specialist requirements management systems to spreadsheets and simple database systems.

Requirements change management

Deciding if a requirements change should be accepted

Problem analysis and change specification

The problem or the change proposal is analyzed to check that it is valid. This analysis is fed back to the change requestor.

Change analysis and costing

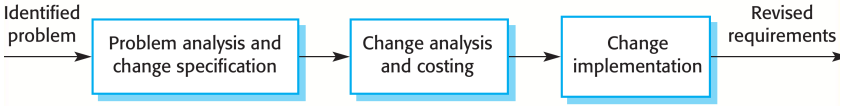
The effect of the proposed change is assessed using traceability information and general knowledge of the system requirements.

Change implementation

The requirements document and, where necessary, the system design and implementation, are modified.

Università degli Studi della Campania *Luigi Vanvitelli*

Requirements change management – process



Key points (1/4)

- 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 (2/4)

- The requirements engineering process is an **iterative process** that includes requirements elicitation, specification and validation.
- Requirements elicitation is an iterative process that can be represented as a spiral of activities – requirements discovery, requirements classification and organization, requirements negotiation and requirements documentation.
- You can use a range of techniques for requirements elicitation including **interviews** and **ethnography**. User stories and scenarios may be used to facilitate discussions.

Università degli Studi della Campania *Luigi Vanvitelli*

Key points (3/4)

- Requirements specification is the process of formally documenting the user and system requirements and creating a **software requirements document**.
- 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.

Key points (4/4)

- **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.

Università degli Studi della Campania *Luigi Vanvitelli*

Grazie per l'attenzione

Prof. Ing. Lelio Campanile, PhD

Assistant Professor

Dipartimento di Matematica e Fisica
Elements of Software Engineering and Information Systems
Corso di Laurea Magistrale in Data Science

Università degli Studi della Campania Luigi Vanvitelli