Requirement engineering

Mariano Ceccato

mariano.Ceccato@univr.it

Requirements

- Descriptions of the features that a system should provide and the constraints on its operation
  - They reflect customer needs for a system that serves a certain purpose
    - E.g., controlling a device, placing an order, or finding information

User/System Requirements

Several contractors can bid for the contract, offering, different ways of meeting the client needs

User requirements

Statements in natural language plus diagrams

Definition of needs in a sufficiently abstract way that a solution is not predefined

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

The client can understands and can validate what the software will do

System requirements

A structured document

System definition for the client in more detail so that the client understands and can validate what the software will do

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

Example: Mentcare

User requirements:

1. The Mentcare system shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System requirements:

1. On the last working day of each month, a summary of the drugs prescribed, their cost and the prescribing clinics shall be generated
2. The system shall generate the report for printing after 17.30 on the last working day of the month
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
4. If drugs are available in different dose units (e.g. 10mg, 20mg, etc.) separate reports shall be created for each dose unit
5. Access to drug cost reports shall be restricted to authorized users as listed on a management access control list.

Stakeholders

- Anyone who is affected by the system in some way
- Anyone who has a legitimate interest in the system
  - E.g., end-users, managers, external regulators who certify the system

Example:

1. Patients whose information is recorded in the system
2. Doctors who are responsible for assessing and treating patients
3. Nurses who coordinate the consultations with doctors and administer treatments
4. Medical receptionists who manage patients' appointments.
5. IT staff who are responsible for installing and maintaining the system.
6. A medical ethics manager who must ensure that the system meets current ethical guidelines for patient care.
7. Health care managers who obtain management information from the system
8. 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.

Functional & non-functional requirements

Functional requirements

- Features and services that the system should provide
- How the system should react to particular input
- How the system should behave in particular situations

più astratta

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

più dettagliata

Different level of abstraction (1 vs 2)

Ambiguous →Different interpretations

Doctor intention: search means in all the clinics

ambiguità

Developer intention: search means in a single clinic. Doctor chose a clinic before searching (easier to implement)

Functional requirements

Objectives:

- Complete: all services and information required by the user should be defined
- Consistent: requirements should not be contradictory

In practice:

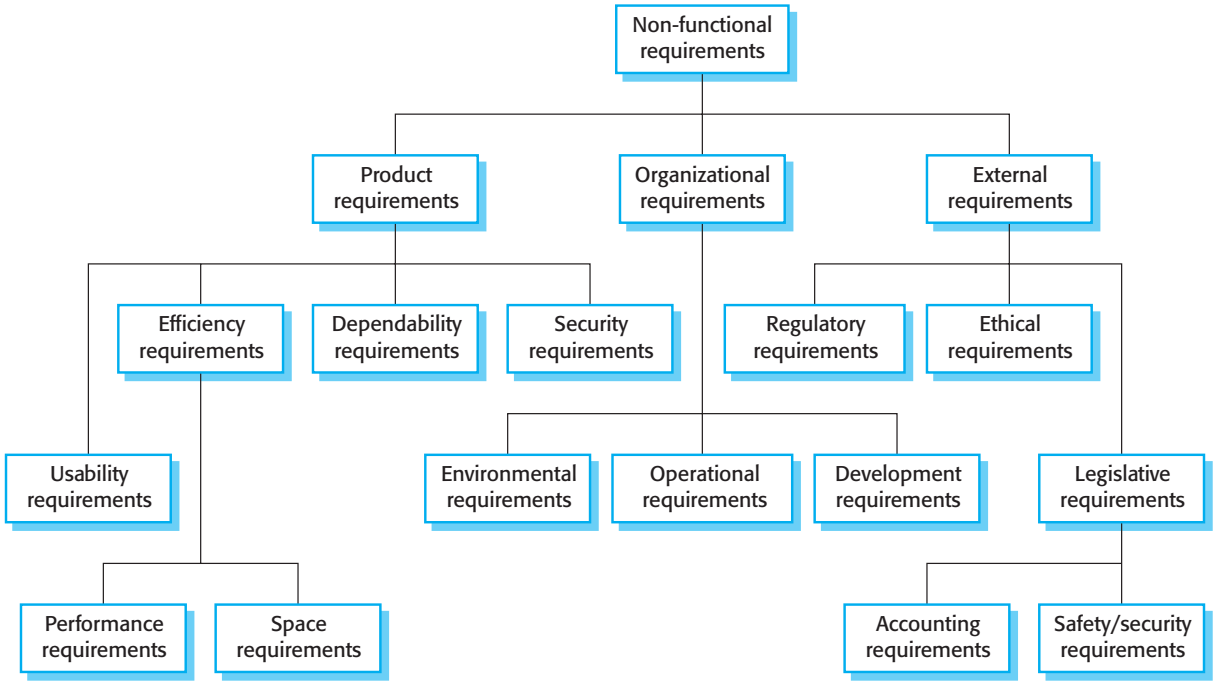
- Stakeholders are likely to have different (often inconsistent) needs
- Imprecision causes disputes between customers and developers
- Inconsistencies may only be discovered after deeper analysis (during development)

Non-functional requirements

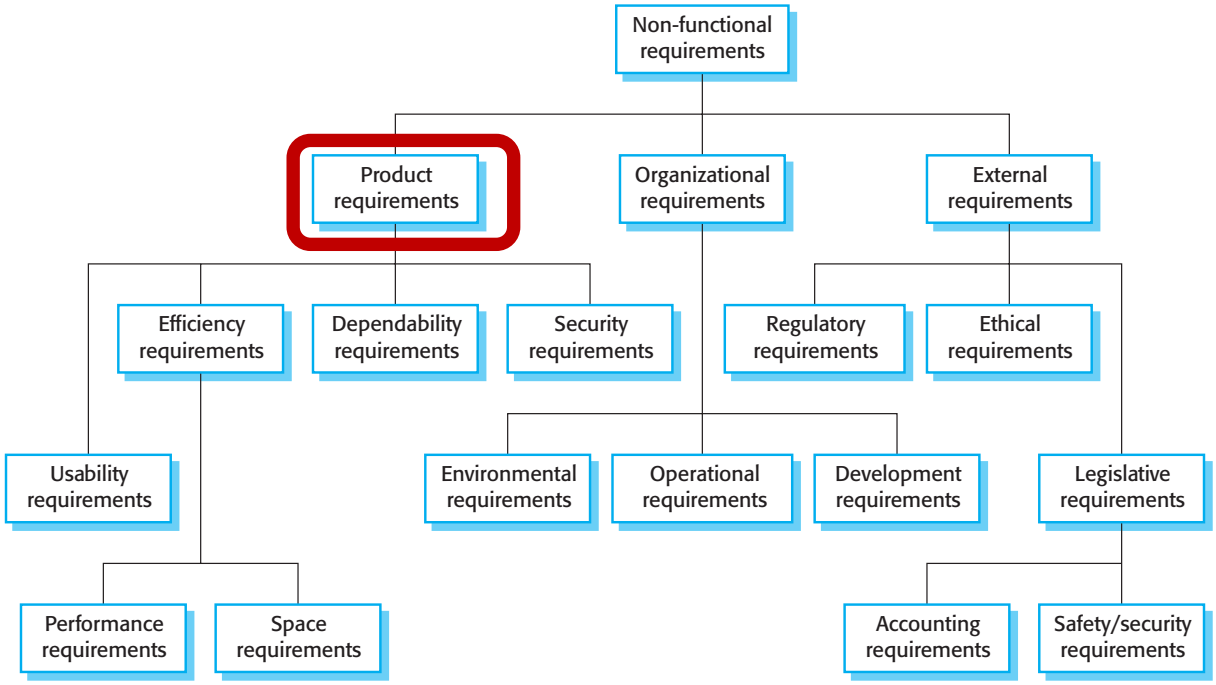
- Constraints on the services or functions offered by the system
- Often apply to the system as a whole, rather than individual features or services.
  - System constraints:
    - Response time, I/O device capability, system representations
  - Development process:
    - Imposing a particular IDE, programming language or development method
  - Standards to meet

1. may affect the overall architecture of a system rather than the individual components
   - E.g., to enforce expected performance in an embedded system, the whole system may need to be organized (to minimize communications among components)
2. A single non-functional requirement might generate several functional requirements

Types of non-functional requirement

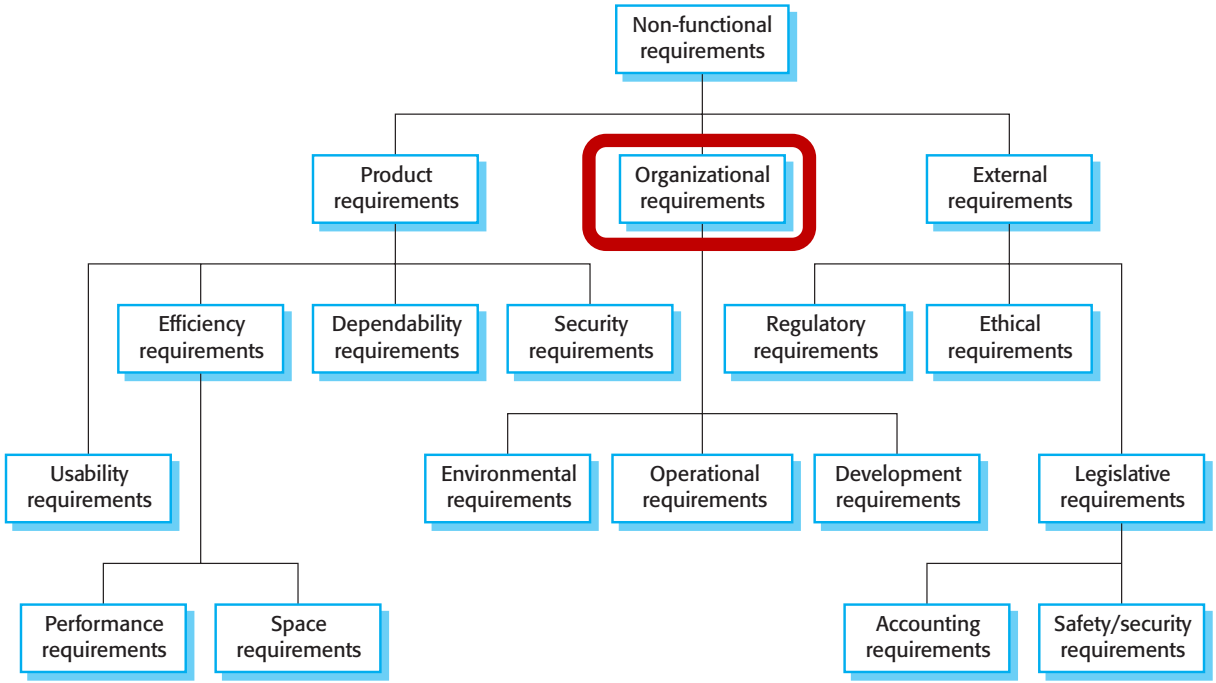


Types of non-functional requirement



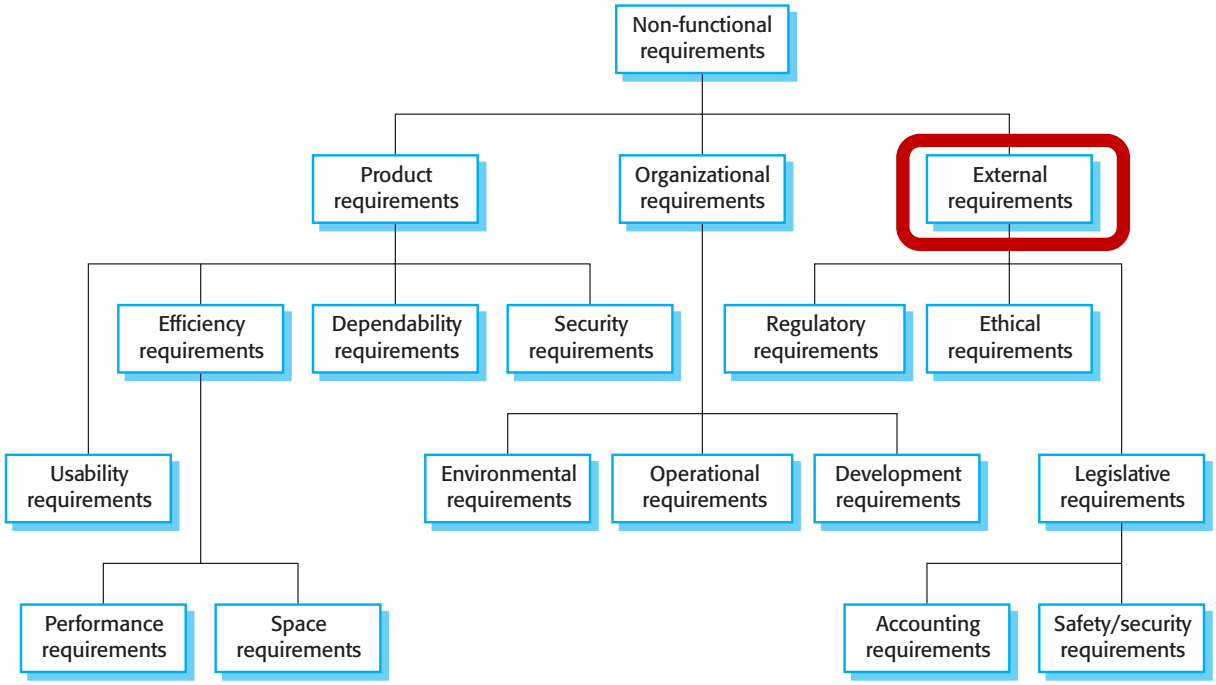
- Product requirements: specify that the delivered product must behave in a particular way:
  - E.g., performance requirements for how fast the system must execute, how much memory it requires;
  - Reliability requirements that set out the acceptable failure rate; security requirements; usability requirements.

Types of non-functional requirement



- Organizational requirements: broad requirements derived from policies and procedures in the customer's and developer's organizations.
  - E.g., operational process requirements on how the system will be used;
  - Development process requirements that specify the programming language; process standards to be used; and the operating environment of the system.

Types of non-functional requirement



- External requirements: derived from factors external to the system and its development process.
  - Regulatory requirements to be met for the system to be approved by a regulator (e.g., nuclear safety authority)
  - Legislative requirements to ensure that the system operates within the law
  - Ethical requirements that ensure that the system will be acceptable to its users and the general public

Example - Mentcare

Product requirement

- The Mentcare system shall be available to all clinics during normal working hours (Mon–Fri, 08:30–17:30). Downtime within normal working hours shall not exceed 5 seconds in any one day.

Organizational requirement

- Users of the Mentcare system shall identify themselves using their health authority identity card.

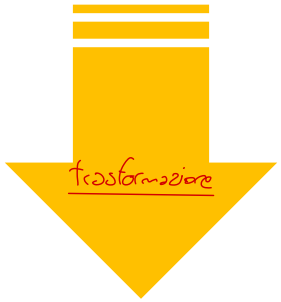
External requirement

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

Verifiable non-functional requirements

- Generic goals cause problems to system developers because leave scope for interpretation and subsequent dispute once the system is delivered.
- When possible, non-functional requirements should be written quantitatively, so they can be objectively tested

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



importante cambiare questo requisito da qualitativo a quantitativo

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

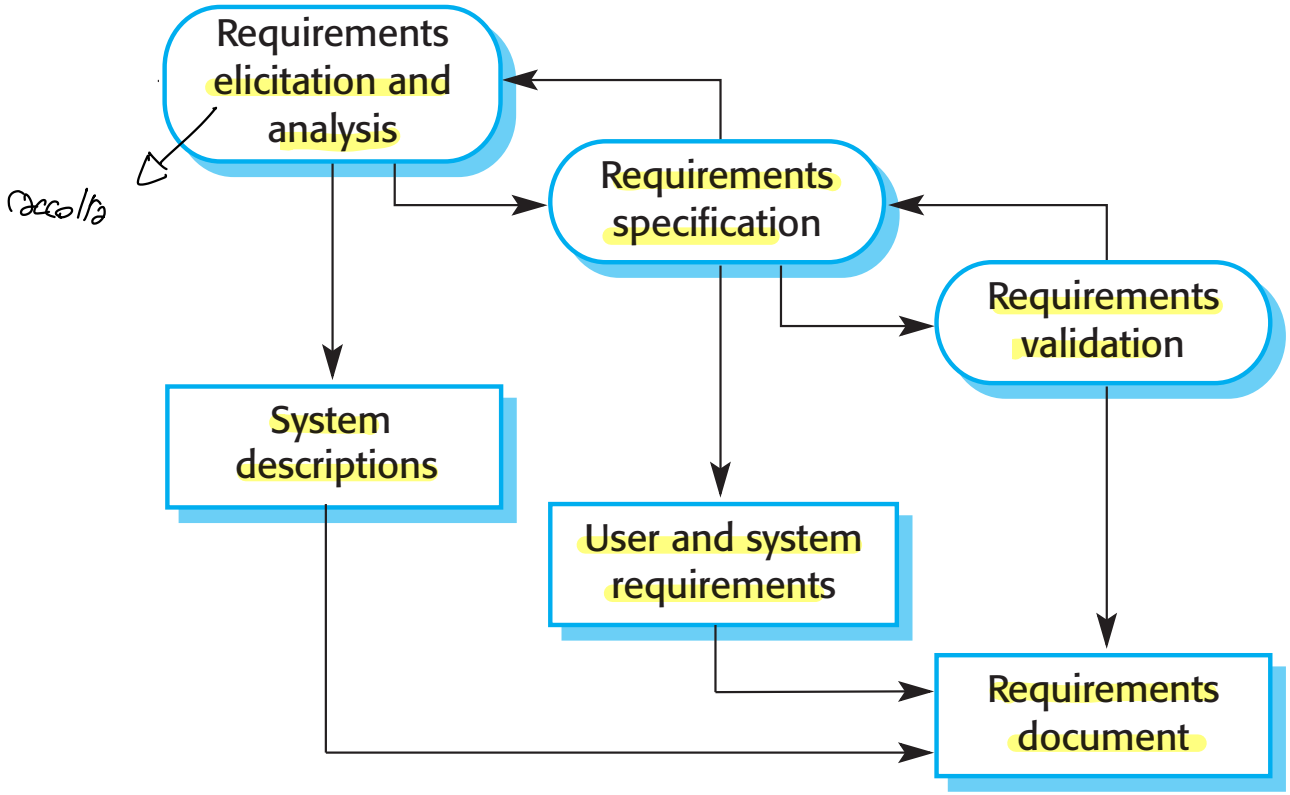
Metrics for non-functional req.

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

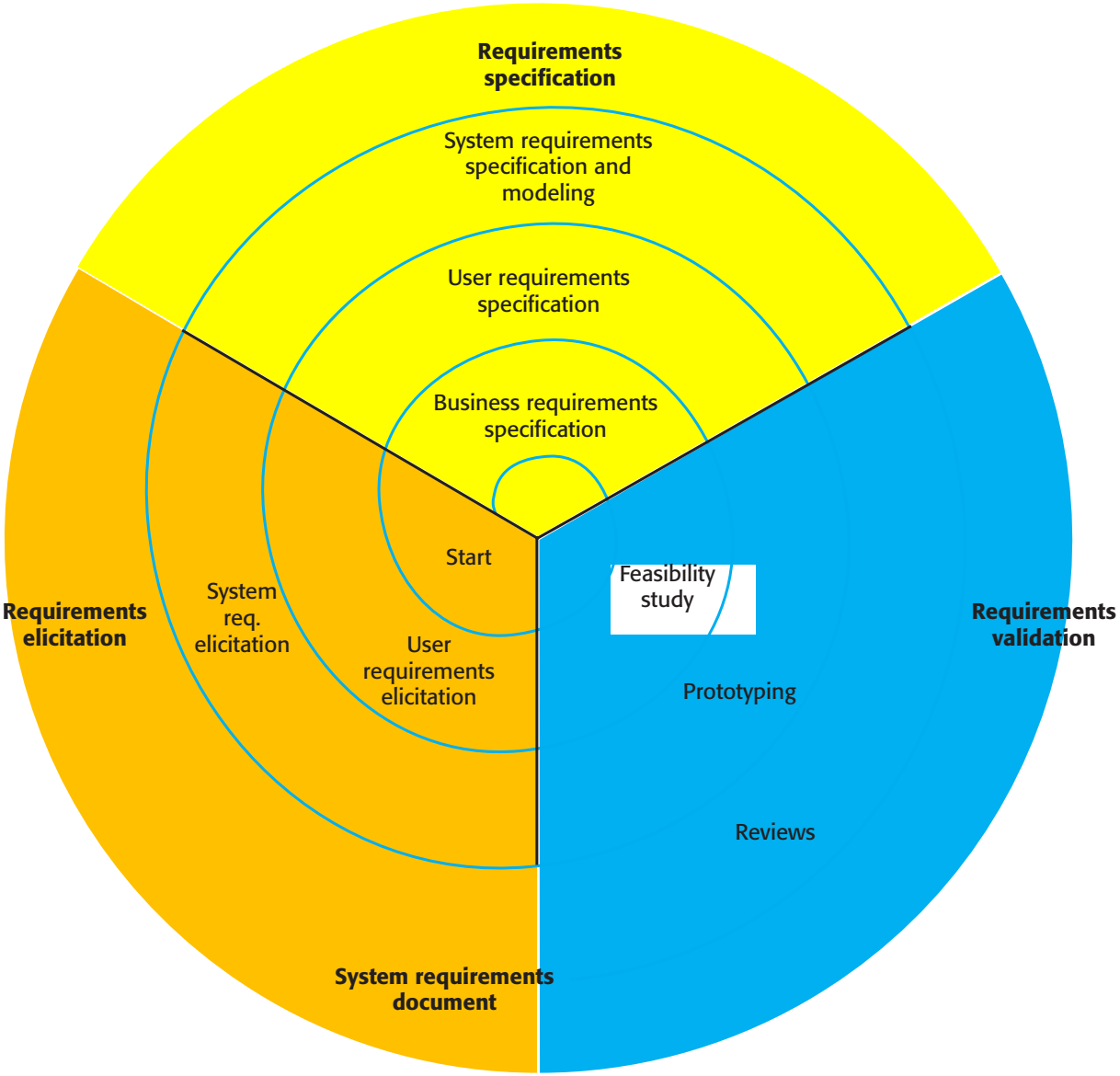
Requirements engineering processes

↳ gestione dei requisiti

Recall from “process” lecture



Requirement engineering process



1. Requirements Elicitation

Requirement elicitation

- **Objective**: understand the work that stakeholders do and how they might use a new system to help support that work

- Stakeholders and software engineers work together to understand:
  - the application domain,
  - the services that the system should provide,
  - the required system performance,
  - hardware constraints, other systems, etc.

Obstacles

- Stakeholders do not know what they really want, or just in very general terms
  - Requests could be infeasible to realize
- Stakeholders express requirements in their own terms
  - Implicit knowledge
- Different stakeholders may have conflicting requirements and different points of views
  - If some stakeholders feel that their views have not been properly considered, they may deliberately undermine the RE process
- Organisational and political factors may influence the system requirements
  - Managers might formulate requirements that help them in gaining more control on the organization
- The requirements change during the analysis process.
  - New stakeholders may emerge
  - The business environment may change.

Interviews

- Types of interview
  - Closed interviews based on pre-determined list of questions
  - Open interviews where various issues are explored with stakeholders, nothing is predefined
  - Mix: answers open new problems, to be discussed in a less structured way
- Stakeholders do not formulate specific and detailed requirements (unless a prototype is shown)

Effective interviewing

- Problems:
  - Domain specific jargon and terms, not understandable by requirements engineers
  - Some concepts might be so familiar to stakeholders that they do not feel the need to clarify them (or are just unable to)
  - Stakeholders might be reluctant to discuss internal organizational (or political) problems

- Suggestions
  - 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
    - Do not start with “tell me what you want”

Ethnography

Lavorare con le persone

People know their work, but not the relation with the rest of the work in their organization

Aim: try to understand the social and organizational issues that affect the use of the system

- Observational technique to understand operational processes and help derive requirements for software to support these processes
  - An analyst immerses himself or herself in the working environment where the system will be used
  - The day-to-day work is observed
  - This helps in discovering implicit requirements
    - Actual ways that people work, rather than the formal processes defined by the organization

Limitation: Ethnography is effective for understanding existing processes but cannot identify new features that should be added to a system.

Stories & Scenarios

- Easier to relate to real-life examples than abstract descriptions
- Description of how a system may be used for a particular task and practical situation
- Stakeholders comment on their situation with respect to a story/scenario

Story:

- Narrative text
- High level description of now the system is sued
- Generic and vast audience
- Anyone can relate to a story

Scenario:

- Structured
- Specific information (input/output)
- Example of interaction among users

A part of a story can then be developed in more detail and represented as a scenario.

Example: a story

Photo sharing in the classroom

Jack is a primary school teacher in Ullapool (a village in northern Scotland). He has decided that a class project should be focused on the fishing industry in the area, looking at the history, development, and economic impact of fishing. As part of this project, 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. Pupils use an iLearn wiki to gather together fishing stories and SCRAN (a history resources site) to access newspaper archives and photographs. However, Jack also needs a photo-sharing site because he wants pupils to take and comment on each other's photos and to upload scans of old photographs that they may have in their families.

Jack sends an email to a primary school teachers' group, which he is a member of, to see if anyone can recommend an appropriate system. Two teachers reply, and both suggest that he use KidsTakePics, a photo-sharing site that allows teachers to check and moderate content. As KidsTakePics is not integrated with the iLearn authentication service, he sets up a teacher and a class account. He uses the iLearn setup service to add KidsTakePics to the services seen by the pupils in his class so that when they log in, they can immediately use the system to upload photos from their mobile devices and class computers.

Example: a scenario

definizione strutturata di come il sistema dovrebbe implementare la funzionalità

Uploading photos to KidstakePics

Initial assumption: A user or a group of users have one or more digital photographs to be uploaded to [illegible] a laptop computer. Users have successfully logged on to KidsTakePics.

A description of the normal flow of events in the scenario

A description of what the system and users expect when the scenario starts

Normal: The user chooses to upload photos [illegible] the computer and to select the project nam[illegible]. The user may 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 [illegible]

A description of what can go wrong and how resulting problems can be handled

[illegible]tomatically sends an email to the project moderator, [illegible]erates an on-screen message to the user that this [illegible]

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. Users should be informed of a possible delay in making their photos visible.

Photos with the same name have already been uploaded by the same user. The user should be asked if he or [illegible]e name, rename the photos, or cancel the upload. I[illegible]ginals are overwritten. If they choose to rename t[illegible]ed by adding a number to the existing filename.

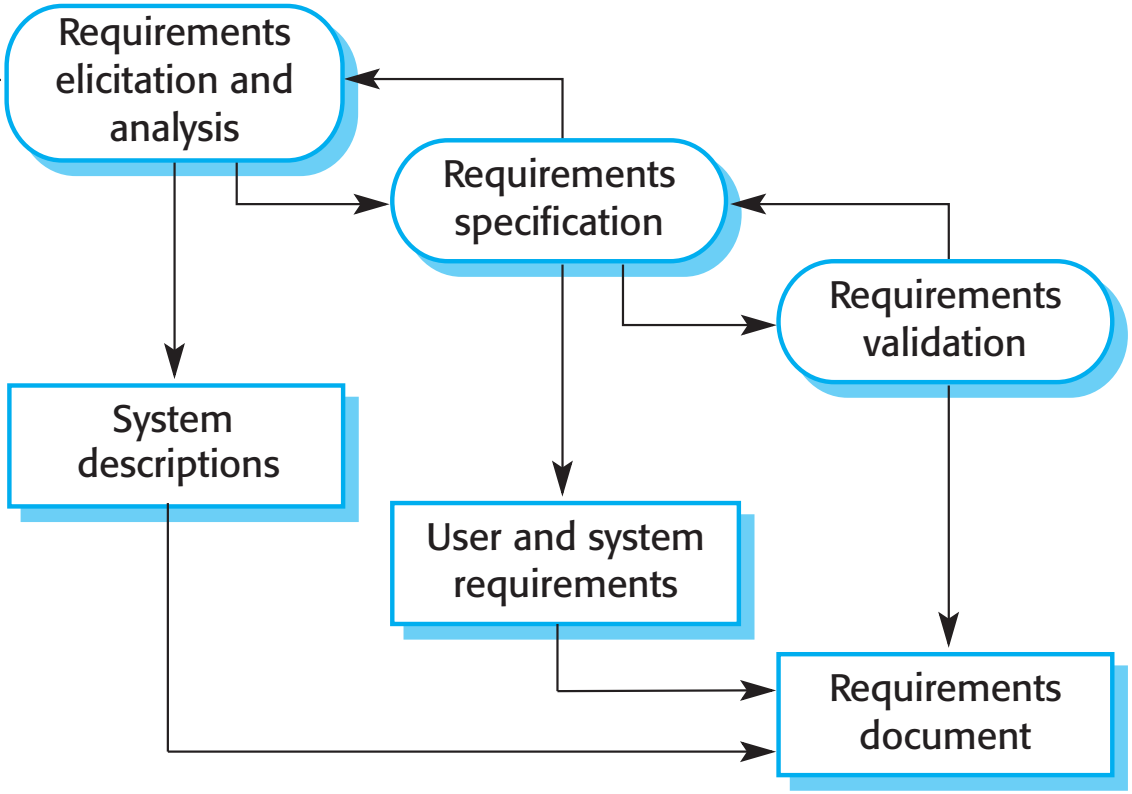
Information about other activities that might be going on at the same time

Other activities: The moderator may be logge[illegible] hey are uploaded.

A description of the system state when the scenario ends

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.

Recall from “process” lecture



2. Requirements specification

Requirement 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 have no 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

- In principle, requirements should state what the system should do and the design should describe how it does this
- In practice, requirements and design could be 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.

Req. specification in natural language

- Natural language is expressive, intuitive, and universal
- But could also be vague and ambiguous
  - interpretation depends on the background of the reader

Guidelines:

- 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 (bold, underline, italics)
- Avoid the use of computer jargon.
- Include an explanation (rationale) of why a requirement is necessary
  - You know whom to consult if the requirement has to be changed

Example: insulin pump

3.2 The system shall measure the blood sugar and deliver insulin, if required, every 10 minutes. *(Changes in blood sugar are relatively slow so more frequent measurement is unnecessary; less frequent measurement could lead to unnecessarily high sugar levels.)*

3.6 The system shall run a self-test routine every minute with the conditions to be tested and the associated actions defined in Table 1. *(A self-test routine can discover hardware and software problems and alert the user to the fact the normal operation may be impossible.)*

Structured req. specification

- Limited freedom of the requirements writer
- Requirements are written in a standard way
- This works well for some types of requirements e.g. requirements for embedded control system
- Sometimes too rigid for writing business system requirements

Example of structured specification

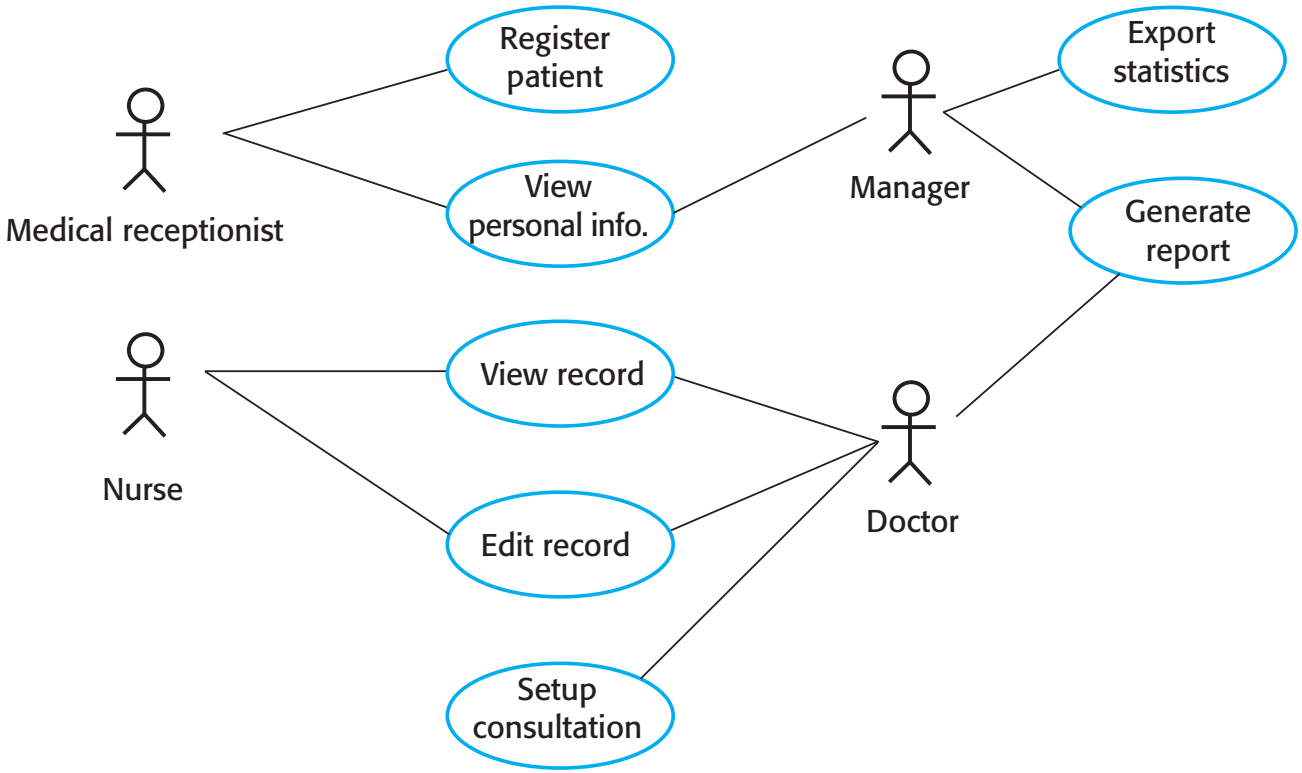
Insulin Pump/Control Software/SRS/3.3.2	
Function	Compute insulin dose: Safe sugar level
Description	Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units
Inputs	Current sugar reading (r2), the previous two readings (r0 and r1)
Source	Current sugar reading from sensor. Other readings from memory
Outputs	CompDose—the dose in insulin to be delivered
Destination	Main cor
Action	CompDo increase is compu and roun that can
Requires	Two prev
Precondition	The insulin reservoir contains at least the maximum allowed single dose of insulin
Postcondition	r0 is replaced by r1 then r1 is replaced by r2
Side effects	None

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) ≥ (r1 – r0))	CompDose = round ((r2 – r1)/4) If rounded result = 0 then CompDose = MinimumDose

Use cases

- Use-cases are a kind of scenario that are included in the UML.
- Use cases identify the actors in an interaction
- 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.

Example: MentCare



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.

Software requirements document

- The official statement of what is required of the system developers
  - Both user requirements and 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
- Variability
  - Information depends on type of system and the approach to development used
  - Systems developed incrementally will, typically, have less detail in the requirements document
  - Standards have been designed (e.g. by IEEE). Standards are mostly applicable to the requirements for large systems engineering projects.

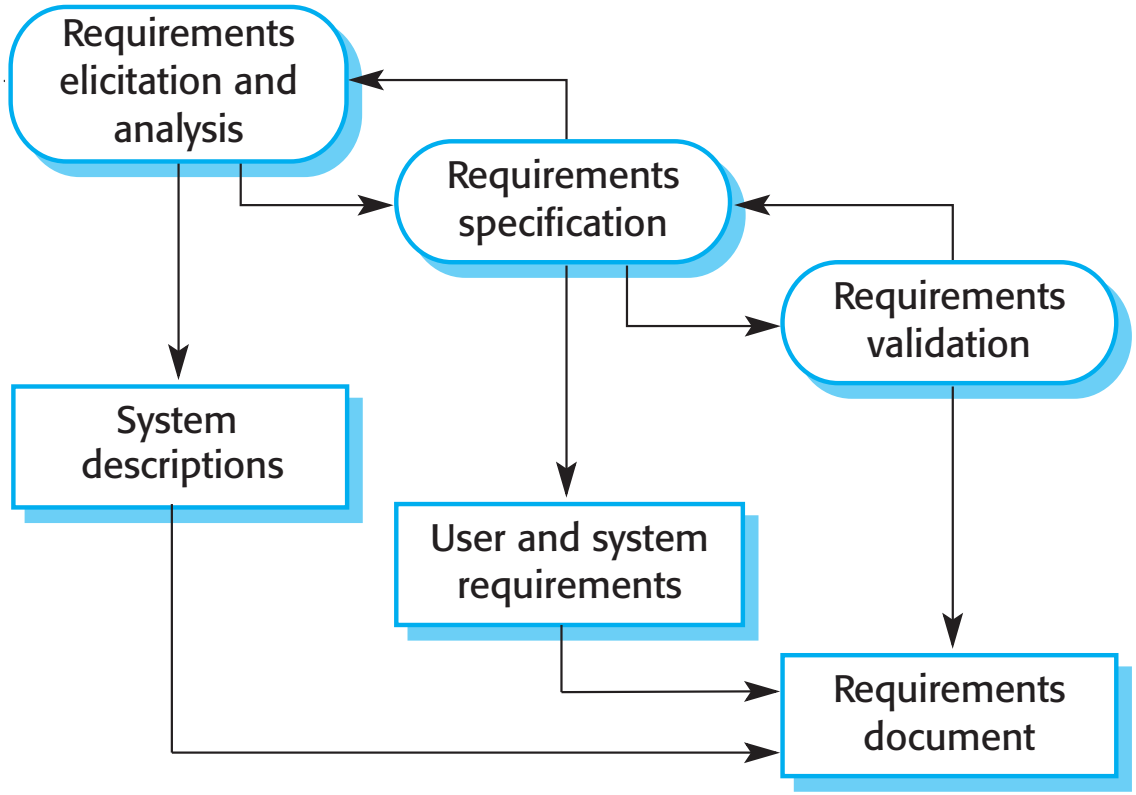
Users of this document

- System customers
  - Specify the requirements and read them to check that they meet their needs
  - Customers 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.

IEEE standard

- Preface
- Introduction
- Glossary
- User requirements definition
- System architecture
- System requirements specification
- System models
- System evolution
- Appendices
- Index

Recall from “process” lecture



3. Requirements validation

Requirements validation

- **Objective**: checking that requirements define the system that the customer really wants
- Very expensive to detect defects in requirements late (x100 scale)
  - Fix design errors
  - Fix implementation errors
  - Validate the changed system

- Checklist:
  - **Validity**: Does the system provide the functions which best support the customer needs?
  - **Consistency**: Are there any requirements conflicts (contradiction, differences)?
  - **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?

Validation techniques

- Requirements reviews
  - Systematic manual analysis of the requirements (auditor team)
- Prototyping
  - Executable model of the system to check requirements with end-users
- Test-cases
  - Developing tests for requirements to check testability
  - If a test is difficult to write, probably the requirement will be difficult to implement

4. Changing requirements

Changing requirements

- In large software systems, requirements are constantly changing
  - Problems intrinsically difficult to be fully defined
  - Changing knowledge of the problem
  - Evolution of technical environment
    - Hardware change
    - Need to interface with other new systems
    - Changing regulations
  - Conflicts in requirements of who *buys* and who *uses* the system
  - Changing priority of different stakeholders.

Requirements management

- Requirements identification: Uniquely identify each requirement (uid), so that it can be cross-referenced with other requirements
- Change management process: Set of activities that assess the impact and cost of changes
- Traceability policies: Relationships between each requirement, and between requirements and the system design
- Tool support: Specialist requirements management systems, or just spreadsheets and simple databases

Change requirement process

- Problem analysis and change specification
  - The problem or the change proposal is analyzed to check that it is valid
  - Discussion with the change requestor for possible revision

- Change analysis and costing
  - The effect and cost of the proposed change is assessed
    - traceability information to estimate the number of changes
  - A decision on whether proceed with the change

- Change implementation
  - The requirements document and, if necessary, the system design and implementation, are modified.
  - The requirement document should be updated, to avoid misalignment with the design/implementation

Course Project

Mariano Ceccato

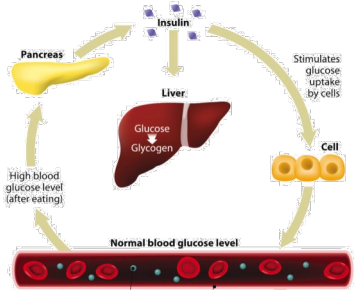
mariano.ceccato@univr.it

Context

- You are the CTO of software development company
- Your company has been assigned a software development contract and you have limited resources (time, budget) to deliver the working software

The software system

- You can chose ONE COMPONENT of these systems:
  - Personal insulin pump (Integrated system)
  - Mentcare (Information system)
  - Wilderness weather station
  - iLearn: A digital learning environment



1. Requirements

- Find informal requirements from lecture notes (incomplete)
- Fill the gaps with your personal interpretation or imagination
- Produce at lest these scenarios

c'è un esempio di scenario nelle slide precedenti

Members in the team	Scenarios
1	5
2	7
3	9