Requirements Engineering

Vũ Thị Hồng Nhạn

(vthnhan@vnu.edu.vn)

Dept. Software Engineering, FIT, UET

Vietnam National Univ., Hanoi

Contents

Requirement 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 **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

- **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

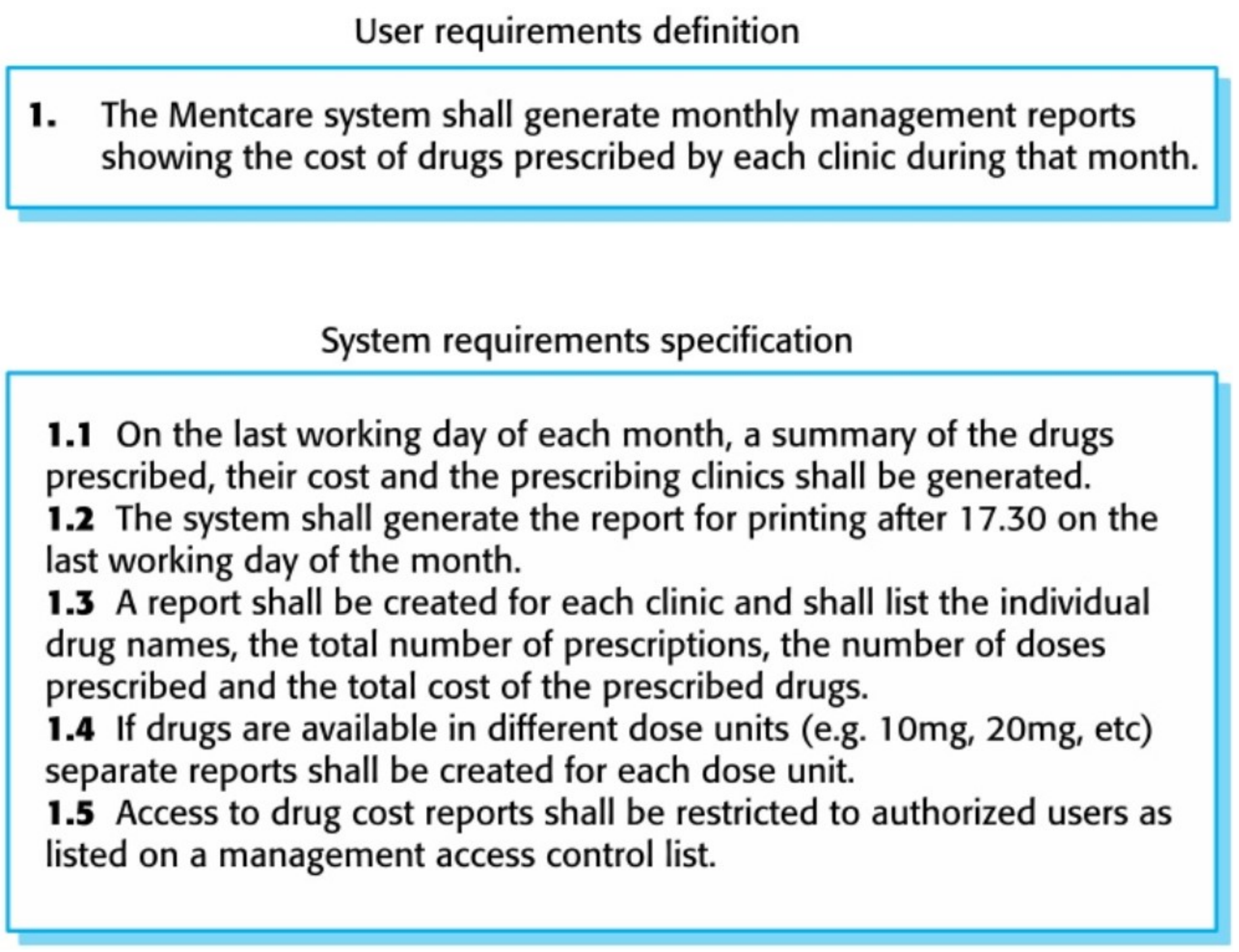
Requirement abstraction (Davis)

- “*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 requirements

- 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 description of the system's functions, services & operational constraints.
  - Defines what should be implemented so may be part of a contract between client & contractor

User & system requirements



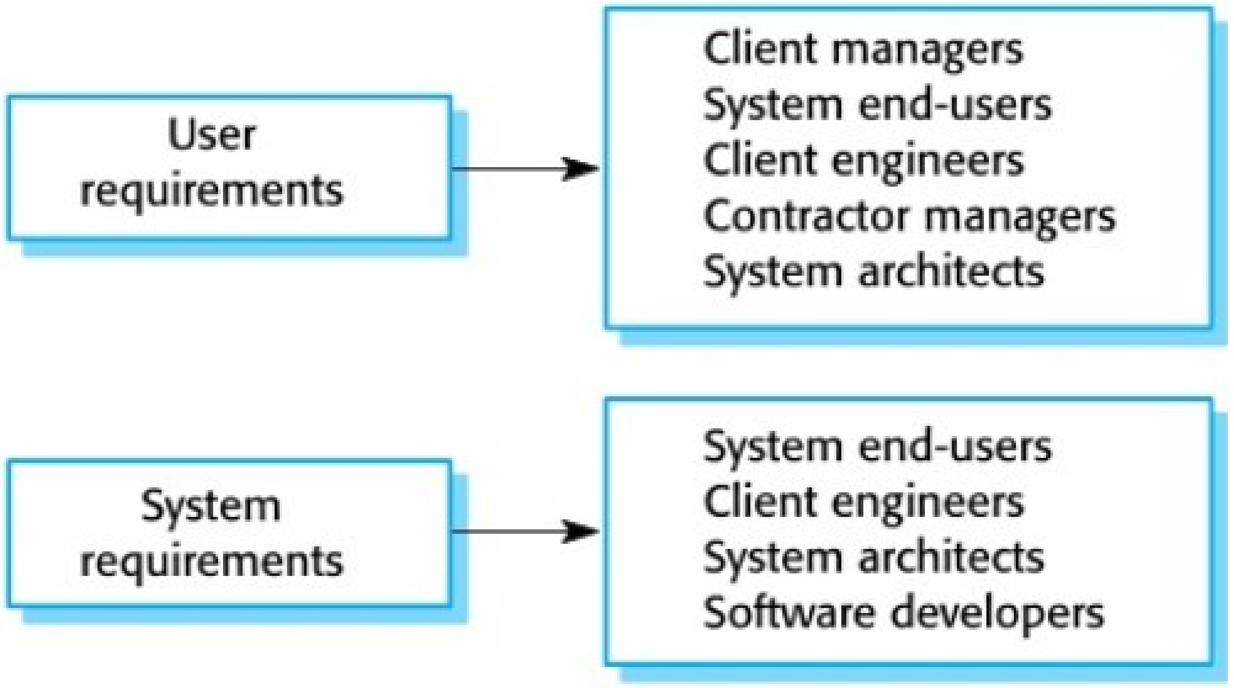
User requirements definition

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



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
  - System owners
  - External stakeholders

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

Stakeholders in the Mentcare 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, preserved, and the record that procedures have been properly implemented

Functional & non-functional requirements

- Functional requirements
  - Statements of services that systems should provide, how the system should react to particular inputs and how the system should behave in particular situations
- Non-functional requirements
  - Constraints on the service 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 that individual features or services

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

Functional requirements

Mentcare system

- **Each staff member** using the system shall be uniquely identified by his or her 8-digit employee number
- **A user** shall be able to search the appointment 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

Requirements imprecision

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

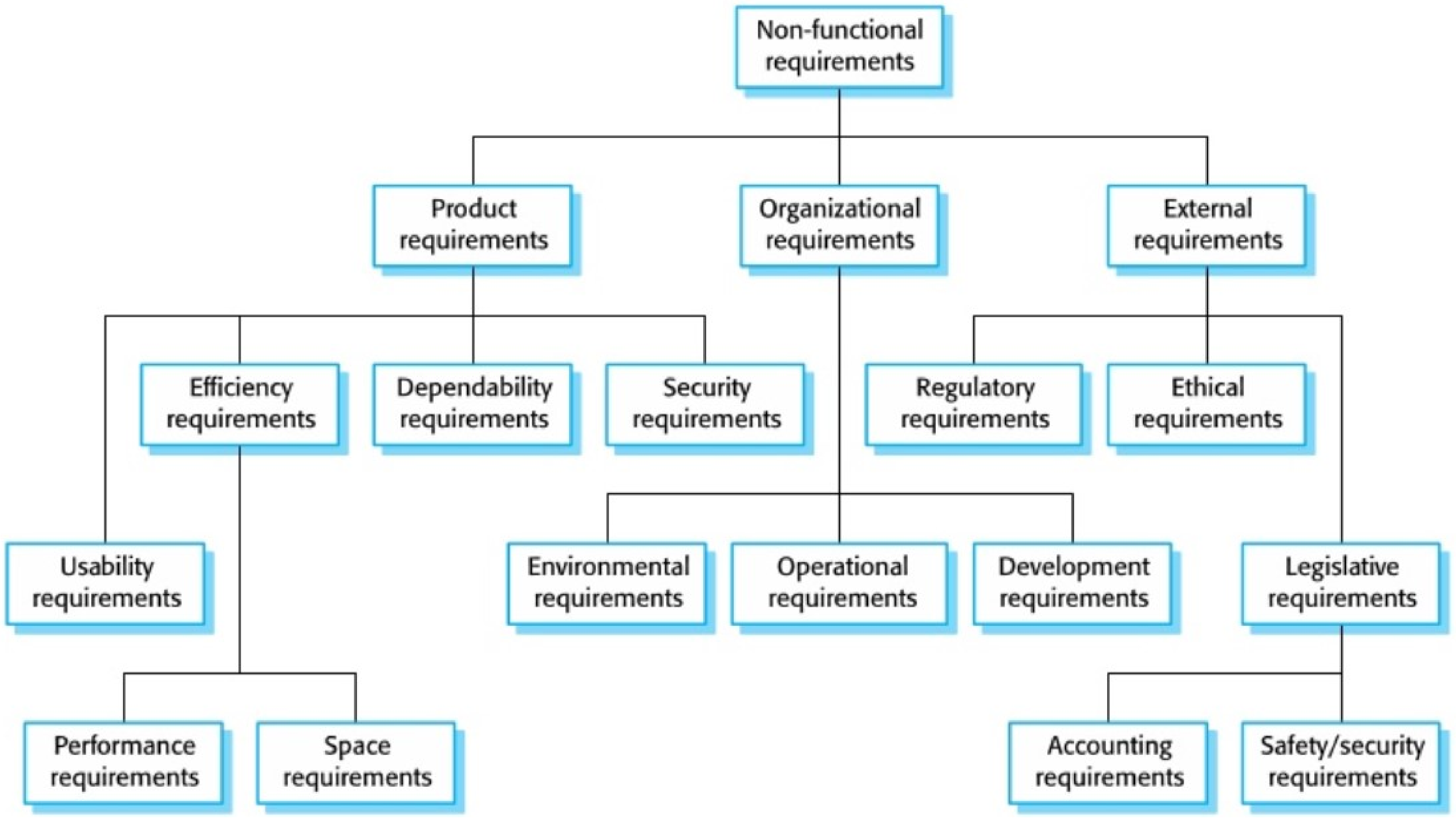
Requirements completeness & consistency

- **In principle**, requirements should be both complete & 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 requirement document

Non-functional requirements

- These define system properties & constraints e.g., reliability, response time & storage requirements. Constraints are I/O device capability, system representation, etc.
- 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

Types of nonfunctional requirement



Non-functional requirements implementation

- Non-functional requirements may affect the overall architecture of a system **rather than** the individual components
  - E.g., to ensure that the performance requirements are met, you may have to organize the system **to minimize** communications between components
- **A single non-functional requirement**, such as security requirement…
  - may generate a number of related functional requirements that define 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.
- Organizational requirements
  - Requirements which are a consequence of organizational 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.

Non-functional classifications

in Mentcare system

- Product requirement
  - The Mentcare system **shall be available** to all clinics during normal working hours (Mon-Fri, 08:30 – 17:30). Downtime within working hours shall not exceed 5 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.

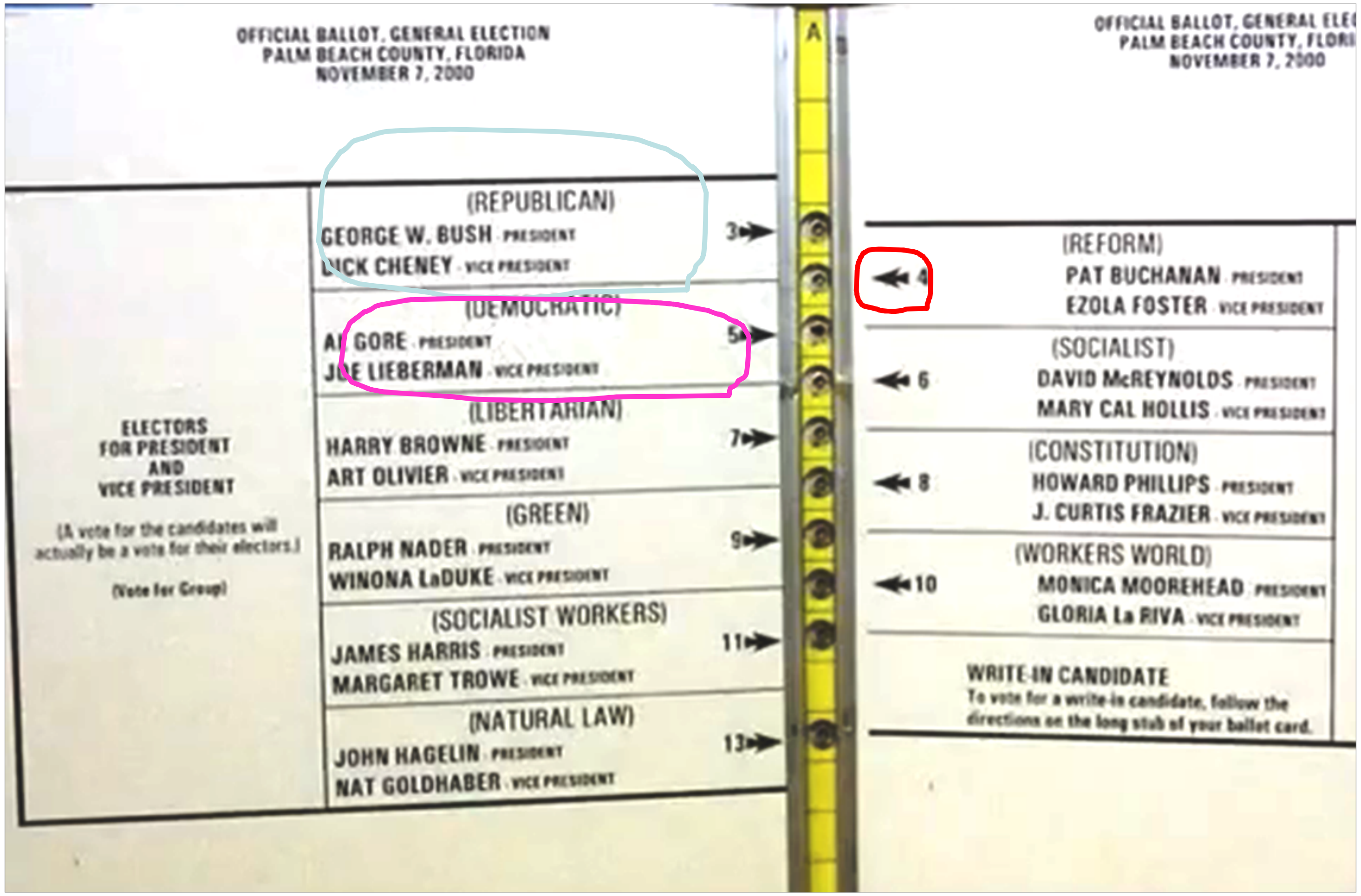
Goals & requirements

- Non-functional requirements may be very difficult to state precisely and imprecise requirements may be difficult to verify
- What are the goals?
  - A general intention of the user such as **ease of use**
  - helpful to developers as they convey the intentions of the system users
- Verifiable non-functional requirement
  - A statement using some measure that can be objectively tested

Usability requirements

- **The system** should be **easy to use** by medical staff and should be organized in such a way that user errors are minimized (Goal)
- Medical staff shall be able to use all the system functions after **4 hours of training**
  - After this training, the average number of errors made by experienced users shall not exceed 2 per hour of system use (Testable non-functional requirement)

E.g.



OFFICIAL BALLOT, GENERAL ELECTION
PALM BEACH COUNTY, FLORIDA
NOVEMBER 7, 2000

ELECTORS
FOR PRESIDENT
AND
VICE PRESIDENT

(A vote for the candidates will actually be a vote for their electors.)

(Vote for Group)

(REPUBLICAN)
GEORGE W. BUSH - PRESIDENT
DICK CHENEY - VICE PRESIDENT
3

(DEMOCRATIC)
AL GORE - PRESIDENT
JOE LIEBERMAN - VICE PRESIDENT
5

(LIBERTARIAN)
HARRY BROWNE - PRESIDENT
ART OLIVIER - VICE PRESIDENT
7

(GREEN)
RALPH NADER - PRESIDENT
WINONA LaDUKE - VICE PRESIDENT
9

(SOCIALIST WORKERS)
JAMES HARRIS - PRESIDENT
MARGARET TROWE - VICE PRESIDENT
11

(NATURAL LAW)
JOHN HAGELIN - PRESIDENT
NAT GOLDHABER - VICE PRESIDENT
13

OFFICIAL BALLOT, GENERAL ELEC
PALM BEACH COUNTY, FLORI
NOVEMBER 7, 2000

4
(REFORM)
PAT BUCHANAN - PRESIDENT
EZOLA FOSTER - VICE PRESIDENT

6
(SOCIALIST)
DAVID McREYNOLDS - PRESIDENT
MARY CAL HOLLIS - VICE PRESIDENT

8
(CONSTITUTION)
HOWARD PHILLIPS - PRESIDENT
J. CURTIS FRAZIER - VICE PRESIDENT

10
(WORKERS WORLD)
MONICA MOOREHEAD - PRESIDENT
GLORIA La RIVA - VICE PRESIDENT

WRITE-IN CANDIDATE
To vote for a write-in candidate, follow the directions on the long stub of your ballot card.

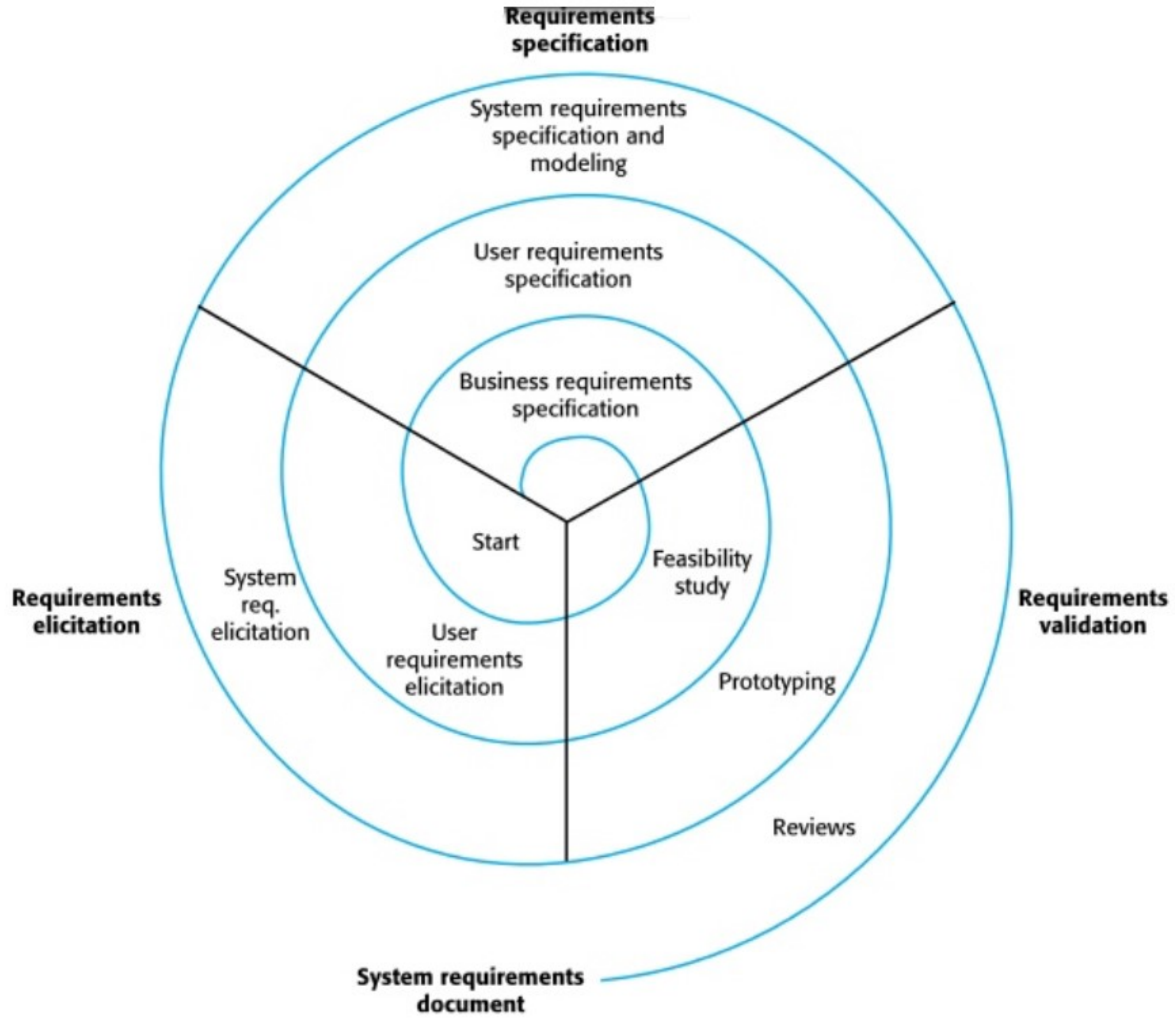
Metrics for specifying nonfunctional requirements

Property	Measure
Speed	Processed transactions/second User/response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frame
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

- **The processes** used for RE **vary widely** depending on the application domain, the people involved and the organization developing the requirements
- However, there are a number of **genetic 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 specification
System requirements specification and modeling
User requirements specification
Business requirements specification
Start
Feasibility study
Requirements elicitation
System req. elicitation
User requirements elicitation
Requirements validation
Prototyping
Reviews
System requirements document

Requirements elicitation & analysis

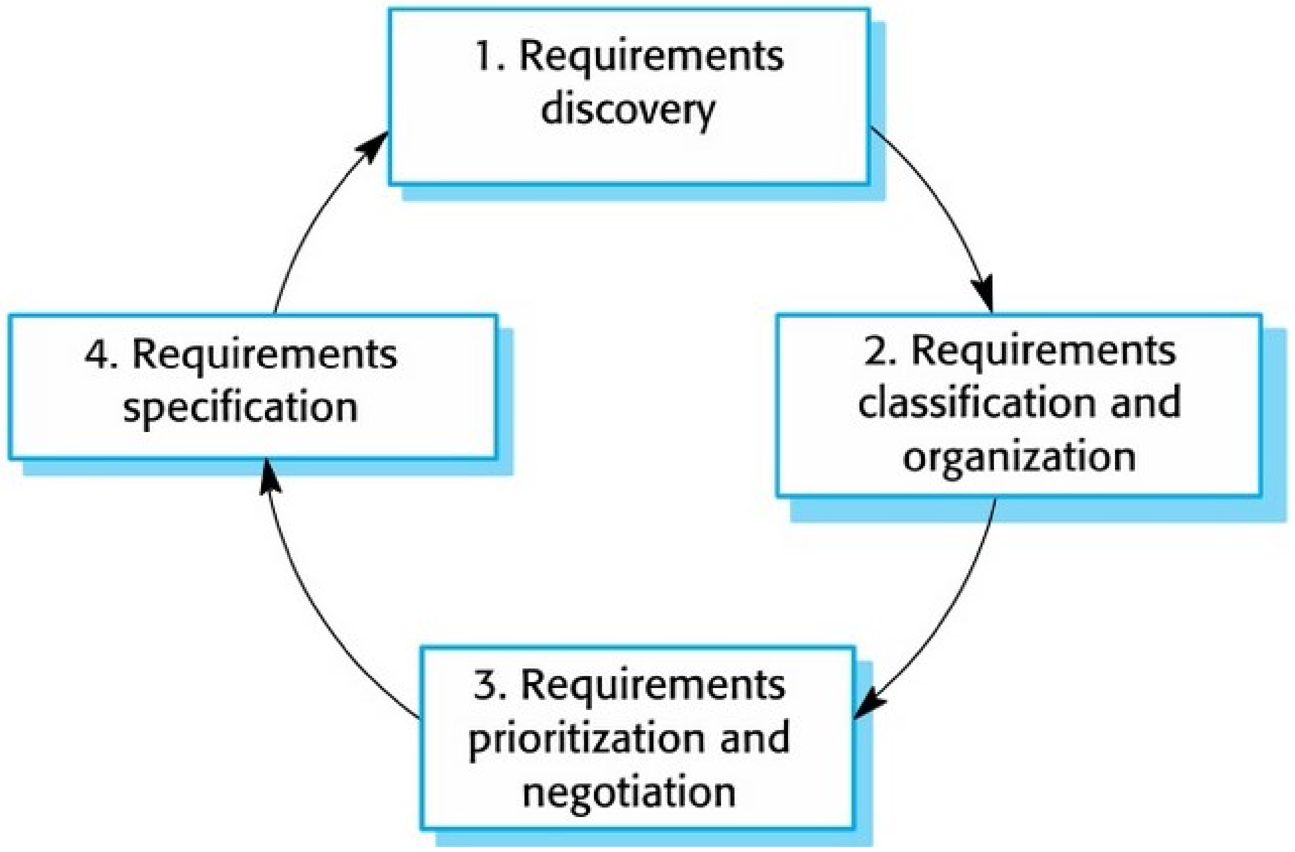
- Sometimes called requirements elicitation or **requirement discovery**
- Involves technical staff working with customers to find out about the application domain, **the services** that the system should provide and the system's **operational constraints**
- May involve end-users, managers, engineers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

←These are called stakeholders

Problems of requirements elicitation

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

The requirements elicitation & analysis process


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

Process activities

- Requirements discovery
  - Interact with stakeholders to discover their requirements
- Requirements classifications and organization
  - Group related requirements
  - Organizes them into coherent clusters
- Prioritization and negotiation
  - Prioritizing requirements and resolving requirements conflicts
- Requirements specification
  - Requirements are documented and input into the next round of the spiral

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 manager 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 and are 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

Interviews with practice

- Normally a mix of closed and open-minded interviewing
- Interviews are good for getting an overall understanding of what stakeholders do and how they might interact with the system
- 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 requirement 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

Ethnography

- **A social scientist** spends a considerable time observing and analyzing how people actually work
- People don't have to explain or articulate their work

- Social and organizational factors of importance may be observed
- **Ethnographic studies** have shown that work is usually richer and more complex than suggested by simple system models

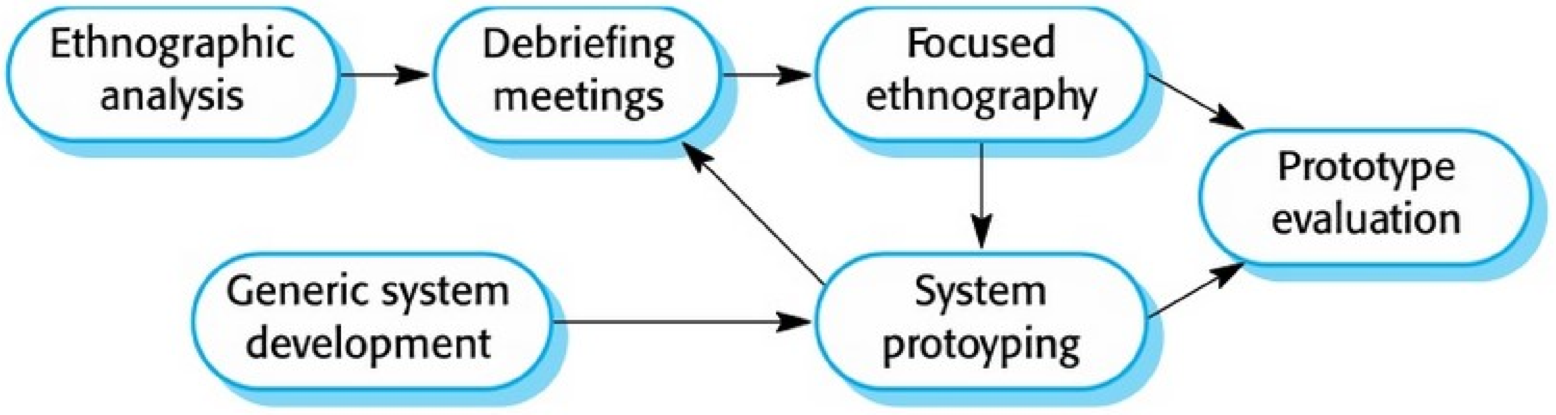
Scope of ethnography

- **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 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

- combines ethnography with prototyping
- Prototype development results in **unanswered questions** which focus on the **ethnographic analysis**
- The problem with ethnography is that it studies existing practices which may have some historical basis which is no longer relevant

Ethnography & prototyping for requirements analysis



Use of stories & scenarios

- **Scenarios** and **user stories** are
  - real-life examples of how a system can be used
  - 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’s 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, students are asked to gather and share reminiscences from relatives, use newspaper archives and collect photographs related to fishing and fishing communities in the area.
- Students use an iLearn wiki to gather fishing stories and SCRAN (a history resources site) to access newspaper archives and photographs.
- However, Jack also needs **a photo sharing site** as he wants students to take and comment on each other’s photos and to upload scans of old photographs that they may have in their families

Photo sharing in the classroom (iLearn)

- 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 uses **KidsTakePics, a photo sharing site** that allows teachers to check authentication service, he **sets up a teacher and a class account.**
- He uses *the iLearn setup service* to add **KidTakePics** to the services seen by the students 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

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)

- **Initial assumption**: a user or a group of users have one ore 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:**
  - The user choose upload photos and they are prompted to select 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 and generates an on-screen message to the user that has been done

Uploading photos

- **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 that there could **be a delay** in making their photos visible
  - Photos with the same name have already been uploaded by the same user. The user should be asked s/he wishes to re-upload the photos with the same name, rename the photos or cancel the upload. If user chose to re-upload the photos, the originals are overwritten. IF user choses to rename the photos, a new name is automatically generated by adding a number to the existing file name
- **Other activities**: the moderator may be logged on the system and may approve photos as they are uploaded
- **System state on completion**: 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

Requirement specification

- The process of writing down the user and system requirements in a **requirement document**

- **User requirements** have to be understandable by end-users and those 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 spec.

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 language	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 specification**
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 requirement 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 the system does

- In practice, **requirements** and **design** are inseparable
  - A system architecture may be designed to structure the requirements
  - **The system** may inter-operate with **other systems** that generate design requirements
  - The use of a specific architecture to satisfy **non-functional requirements** may be a domain requirement

Natural language spec.

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

Guidelines for writing requirements

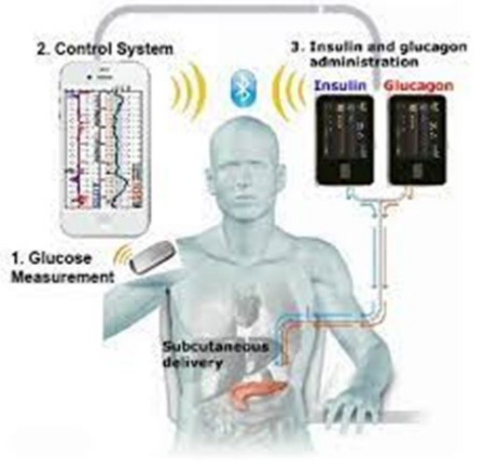
- Invent **a standard format** and use it for all requirements
- Use language in a consistent way. Use **shall** for mandatory requirements, **should** for desirable requirements
- Use text highlighting to identify **key parts** of the requirement
- **Avoid** the use of computer jargon
- Include an explanation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

Problems with natural language

- Lack of clarity
  - Difficult to use language in a precise and unambiguous way without making the document wordy and difficult to read
- Requirements confusion
  - Functional & non-functional requirements tend to be mixed up
- Requirements amalgamation (union)

Example requirements for the insulin pump software system

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 advance. (A self-test routine can discover **hardware** and **software problems** and **alert** the user that the normal operation may be impossible)

Structured spec.

- 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 system, but is sometimes **too rigid** for writing business system requirements

Form-based spec.

- Definition of **the function** or entity
- Description of **inputs** and where they come from
- Description of **outputs** and where they go to
- Information about 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 spec. of a requirement

for an insulin pump

Insulin Pump/Control Software/SRS/3.3.2

- **Function**: Compute insulin dose: safe sugar level
- **Description**
  - Compute the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 & 7 units
- **Input:** Current sugar reading (r2); the previous readings (r0 and r1)
- **Source:**
  - Current sugar reading from sensor. Other readings from memory
- **Output:** CompDose- the dose in insulin to be delivered
- **Destination**: main control loop

A structured spec. of a requirement

for an insulin pump

- **Action**
  - CompDose is zeo if the sugar level is stable or falling or if the level is increasing by 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 & the previous level by 4 and rounding the result.
  - If the result is rounded to 0, then CompDose is set to the minimum dose that can be delivered
- **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 spec.

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

Tabular spec.

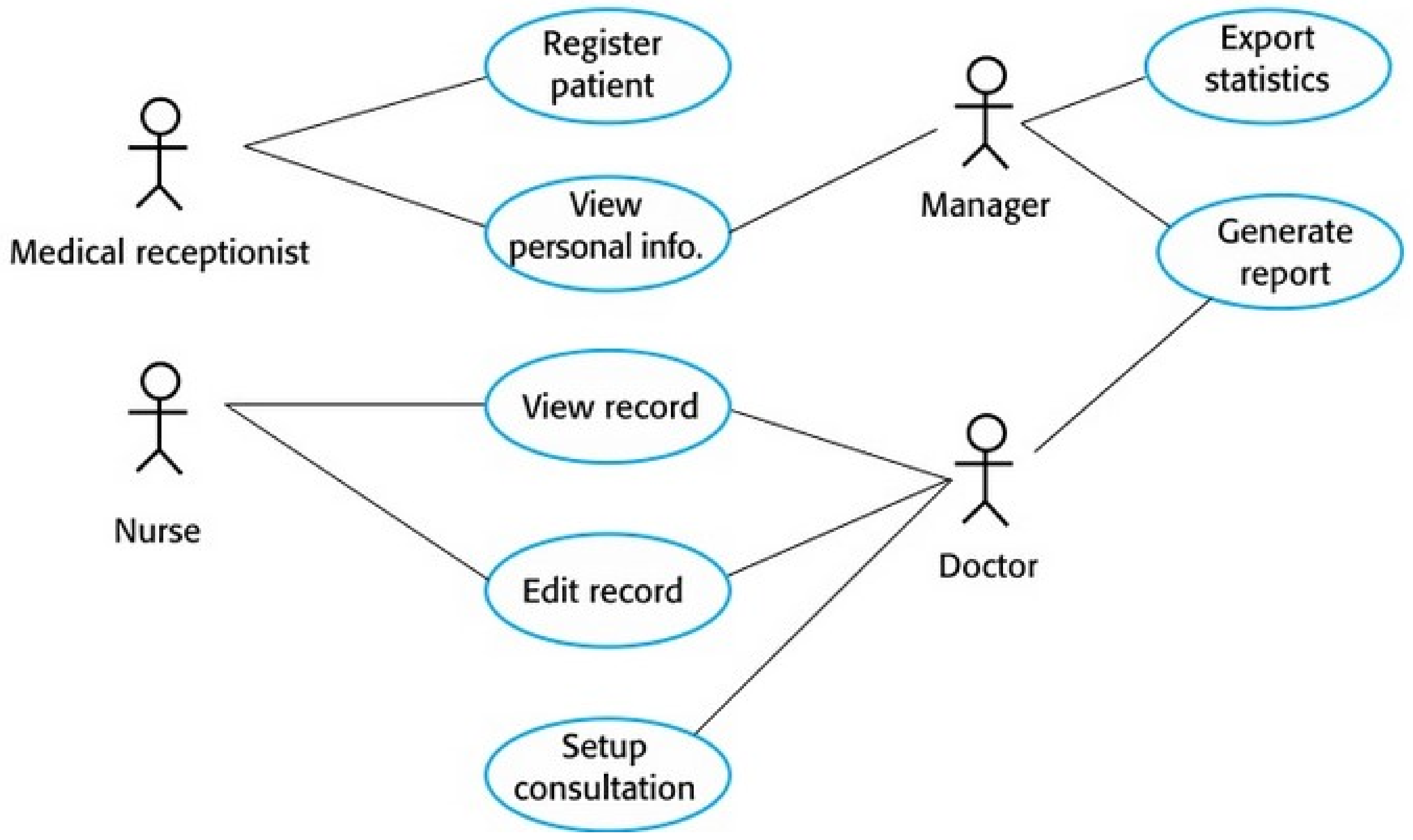
of computation for an insulin pump

Condition	Action
Sugar level falling (r2<r1)	CompDose =0
Sugar level stable (r2=r1)	CompDose =0
Sugar level increasing & rate of increase decreasing (r2-r1) < (r1-r0)	CompDose =0
Sugar level increasing & 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 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

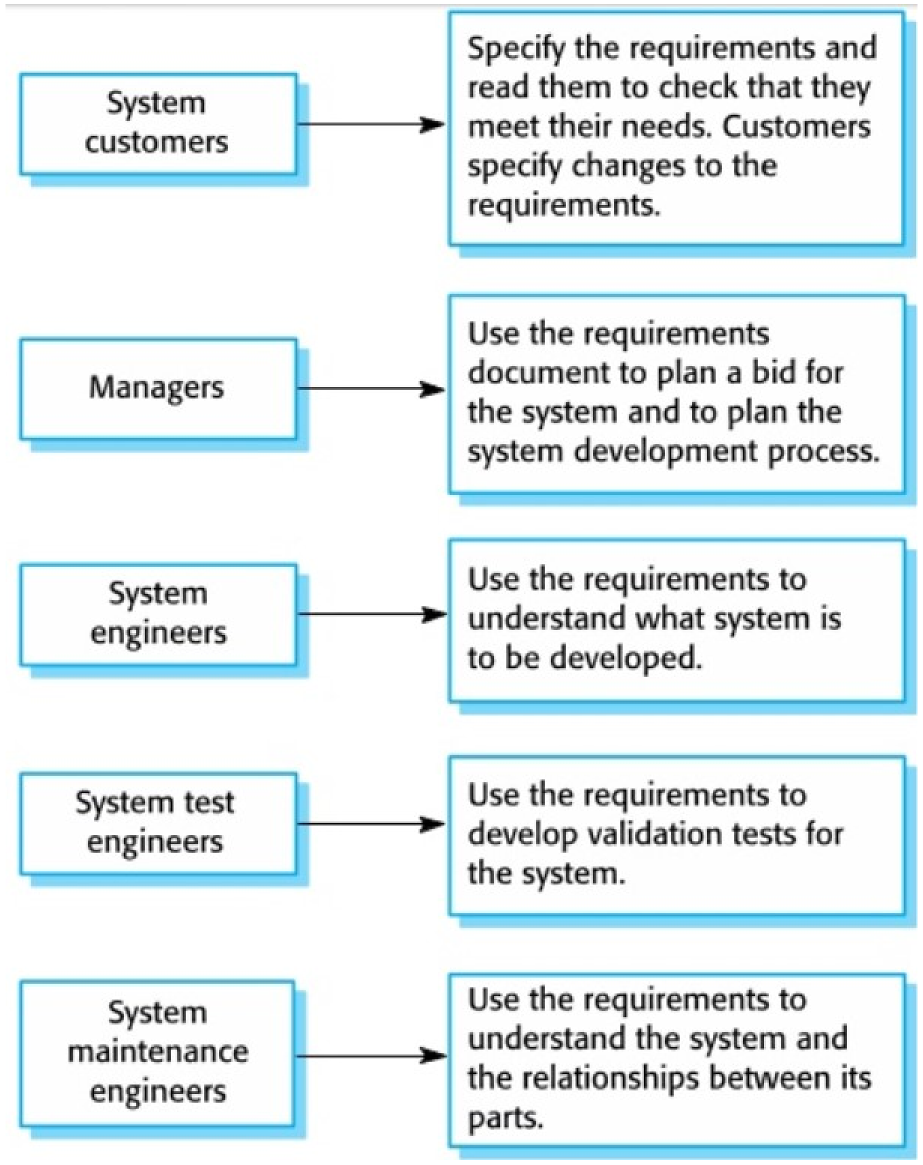
Use cases for the Mentcare system



The software requirements document

- **The software requirements document** is the official statement of what is required from the system being developed
- Should include both a definition of **user requirements** and a specification of **the system requirements**
- It is not a design documents
  - as far as possible, it should be set of **What** the system should do rather than **How** it should do it

Users of a requirement 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.

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

The structure of a requirement 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 systems' 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 specification definition	Here, you describe the services provided for the user. The non-functional 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 requirement document

Chapter	Description
System requirements specification	This should describe the functional & non-functional requirements in more detail. If necessary, further detail may also be added to the non-functional requirements. Interfaces to other systems may be defined
System evolution	This should describe fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on.
Appendices	These should provide detailed, specific information that is related to the application being developed, e.g., hardware & 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 or diagrams, an index of functions, and so on

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 costs up to 100 times the costs of fixing an implementation error

Requirements checking

- Validity
  - Does the system provide the function which best support the customer' needs?
- Consistency
  - Are there any requirements conflicts?
- Completeness
  - Are **all functions** required by the customer included?
- Realism
  - Can the requirements be implemented **given** *available budget* and *technology*
- Verifiability
  - Can the requirements be checked?

Requirements 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

Requirement reviews

- Regular reviews should be held while the requirements definition is being formulated
- Both client & 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 any 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

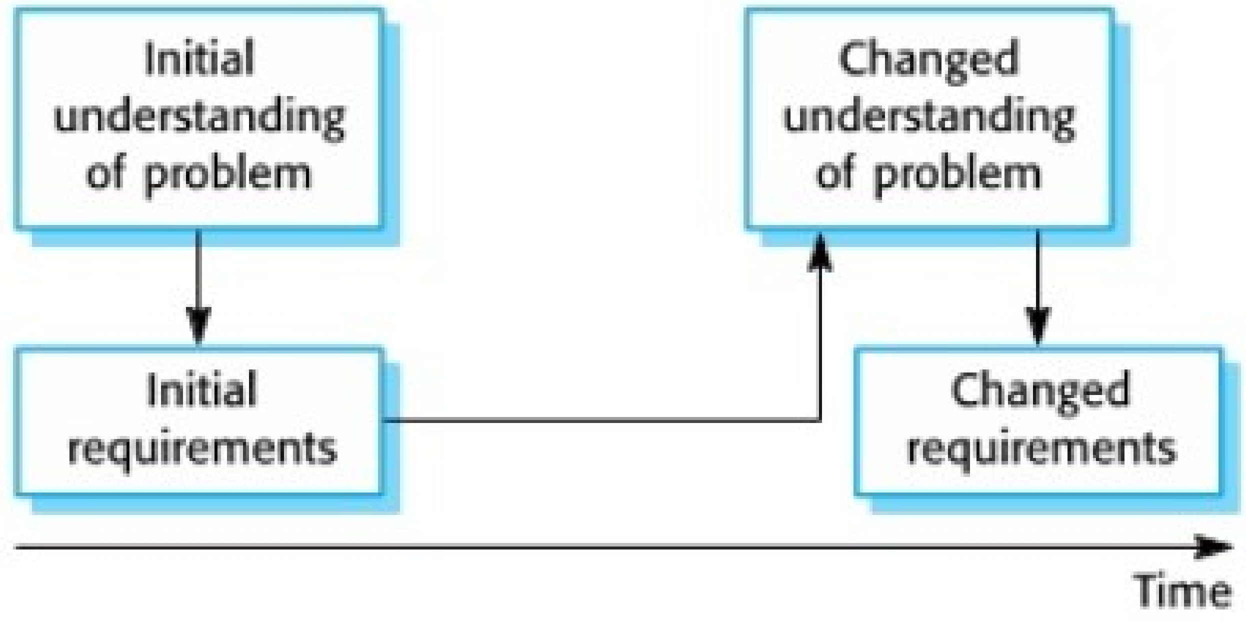
Requirements change

- 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 (with consequent changes in the system support required), and new legislation and regulations may be introduced that the system must necessarily abide by
- 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 and **after** delivery, new features may have to be added for user support

Requirements change

- 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


Initial understanding of problem
Changed understanding of problem
Initial requirements
Changed requirements
Time

Requirements management

- Requirements management is **the process** of managing **requirements changes** during the RE 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 & 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

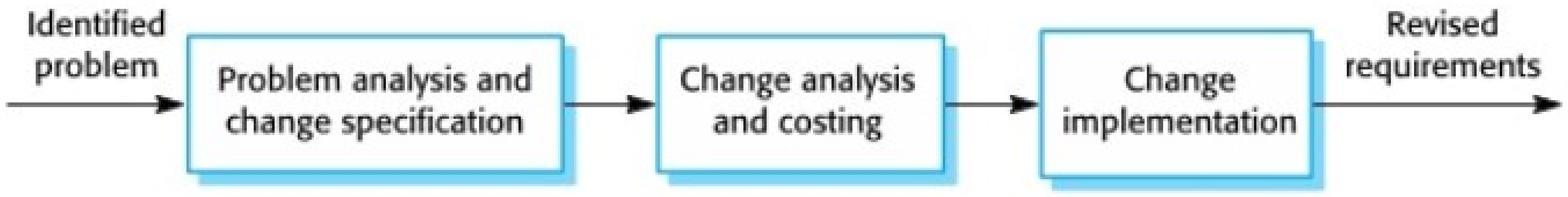
  - **Requirements identification**: each requirement must be uniquely indentified so that it can be cross-referenced with other requirements

  - **A change management process**: This is a set of activities that assess the impact and cost of changes.

  - **Traceability policies**: These policies define the relationships between each requirement and between the requirements and the system design that should be recorded

  - **Tool support**: Tools that may be used range from specialist requirements management systems to spreadsheets and simple database systems

Requirements change management


Identified problem
Problem analysis and change specification
Change analysis and costing
Change implementation
Revised requirements

Key points

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

Key points...

- The requirement engineering process is an iterative process that includes requirements elicitation, specification and validation
- Requirement 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 & scenarios may be used to facilitate discussions

Key points...

- **Requirements specification** is the process of formally documenting the user & system requirements and creating a software requirements documentation
- **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
- Requirement validation is the process of checking the requirements for validity, consistency, completeness, realism and verifiability
- Business, organizational & technical changes inevitability lead to changes to the requirements for a software system. Requirements management is the process of managing and controlling these changes