



## Chapter 3: REQUIREMENTS ENGINEERING (Công Nghệ Yêu Cầu)

PhD. Nguyễn Thị Hạnh

Chapter 3 Requirements engineering

1

### Topics covered

- Functional and non-functional requirements
- The software requirements document
- Requirements specification
- Requirements engineering processes
- Requirements elicitation and analysis
- Requirements validation
- Requirements management

Chapter 3 Requirements engineering

2

## Requirements Engineering

- The process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed.
- The requirements themselves 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 (đấu thầu) 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.

## Requirements Abstraction (Davis)

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

## Case Study

- MHC PMS
- Mental Health Care Patient Management System
- This case study is based on a real system that is in use in a number of hospitals.
- For reasons of commercial confidentiality, I have changed the name of the system and have not included information about any specific system features

## Types of Requirement

- **User requirements** (Yêu cầu người dùng)
  - Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.
- **System requirements** (Yêu cầu hệ thống)
  - 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

### Example

#### - **User requirements definition**

1.1 MHC-PMS sẽ lập báo cáo quản lý hàng tháng thể hiện chi phí thuốc theo chỉ định của từng phòng khám trong tháng đó.

#### - **System requirements specification**

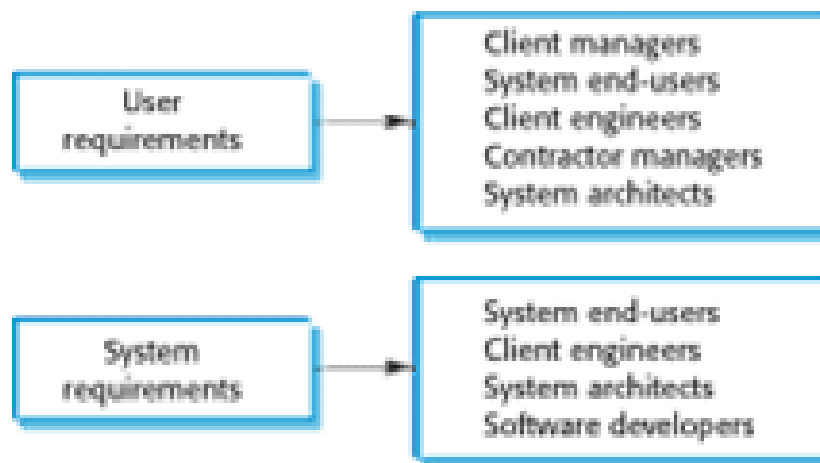
- 1.1 Vào ngày làm việc cuối cùng của tháng, phải lập bảng tổng hợp các loại thuốc được kê đơn, giá thành và cơ sở kê đơn.
- 1.2 Hệ thống sẽ tự động tạo báo cáo để in sau 17h30 ngày làm việc cuối cùng của tháng.
- 1.3. Mỗi phòng khám phải lập báo cáo, trong đó liệt kê tên thuốc, tổng số đơn thuốc, số liều kê và tổng chi phí của từng loại thuốc.
- 1.4 Nếu thuốc có sẵn ở các đơn vị liều khác nhau (ví dụ: 10mg, 20 mg, v.v.) thì phải tạo các báo cáo riêng cho từng đơn vị liều.
- 1.5 Quyền truy cập vào tất cả các báo cáo chi phí sẽ bị hạn chế đối với những người dùng được ủy quyền được liệt kê trong danh sách kiểm soát truy cập của ban quản lý.

## Case Study MHC PMS

Let's give one example

- **User requirements definition**
- **System requirements specification**

## Readers of Requirements Specification



## 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

## 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

### Functional Requirements for the MHC-PMS

- Người dùng có thể tìm kiếm danh sách cuộc hẹn của tất cả các phòng khám.
- Hệ thống sẽ tạo ra mỗi ngày, cho mỗi phòng khám, một danh sách bệnh nhân dự kiến đến thăm khám vào ngày hôm đó.
- Mỗi nhân viên sử dụng hệ thống sẽ được nhận dạng duy nhất bằng mã số nhân viên gồm 8 chữ số của mình.

## Requirements Imprecision (sự mơ hồ)

- Problems arise when 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 chooses clinic then search.

## Requirements Completeness and Consistency

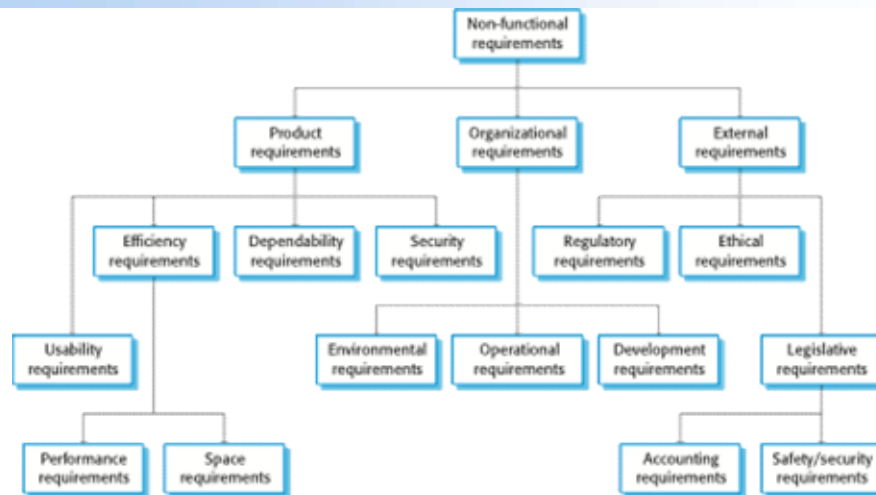
- In principle, requirements should be both **complete** and **consistent**.
- Complete
  - They should include descriptions of all facilities required.
- Consistent
  - There should be no conflicts or contradictions in the descriptions of the system facilities.
- In practice, **it is impossible to produce 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.
- **Non-functional Requirements may be more critical than Functional Requirements. If these are not met, the system may be useless.**



## Types of Non-functional Requirements



Chapter 3 Requirements engineering

17

## 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.

Chapter 3 Requirements engineering

18

## 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.

## Non-functional Classifications

- Examples of Non-functional Requirements in the MHC-PMS

### Product requirements

MHC-PMS sẽ có sẵn cho tất cả các phòng khám trong giờ làm việc bình thường (Thứ Hai – Thứ Sáu, 08:30–17.30). Thời gian ngừng hoạt động trong giờ làm việc bình thường không được vượt quá năm giây trong một ngày bất kỳ.

### Organizational requirements

Người dùng hệ thống MHC-PMS phải tự xác thực bằng thẻ căn cước của cơ quan y tế.

### External requirements

Hệ thống phải thực hiện các điều khoản về quyền riêng tư của bệnh nhân như được quy định trong HStan-03-2006-priv.

## Case Study MHC-PMS

Let's give one example

- **Product Requirements**
- **Organizational requirements**
- **External requirements**

## 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.

## 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 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. (Testable non-functional requirement)

## 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

## Domain Requirements

- The system's operational domain imposes requirements on the system.
  - For example, a train control system has to take into account the braking characteristics in different weather conditions.
- Domain requirements be new functional requirements, constraints on existing requirements or define specific computations.
- If domain requirements are not satisfied, the system may be unworkable.

## Train Protection System

- This is a domain requirement for a train protection system:
- The deceleration of the train shall be computed as:
  - $D_{train} = D_{control} + D_{gradient}$
  - where  $D_{gradient}$  is  $9.81ms^2 \cdot \text{compensated gradient}/\alpha$  and where the values of  $9.81ms^2 / \alpha$  are known for different types of train.
- It is difficult for a non-specialist to understand the implications of this and how it interacts with other requirements.

## Domain Requirements Problems

- Understandability
  - Requirements are expressed in the language of the application domain;
  - This is often not understood by software engineers developing the system.
- Implicitness
  - Domain specialists understand the area so well that they do not think of making the domain requirements explicit.



## Software Requirements Document



## Software Requirements Document

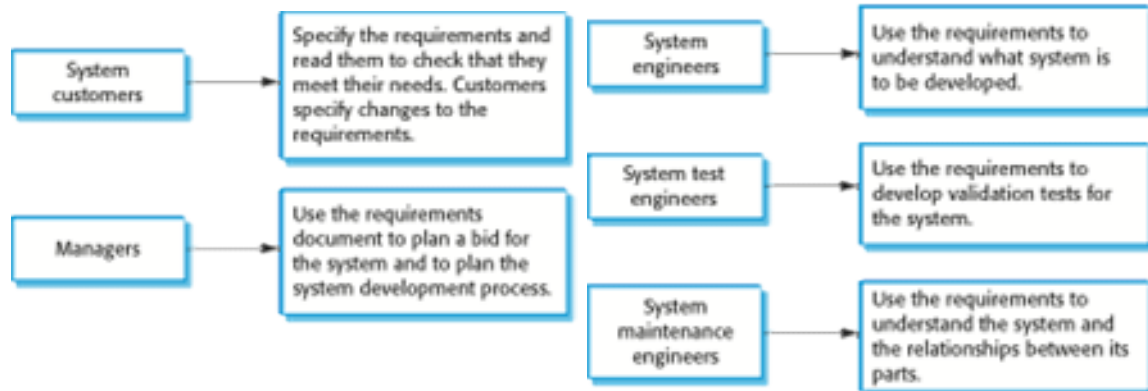
- The software requirements document is the official statement of what is required of the system developers.
- Should include both a definition of user requirements and a specification of the system requirements.
- It is NOT a design document. As far as possible, it should set of WHAT the system should do rather than HOW it should do it.



## Agile Methods and Requirements

- Many agile methods argue that producing a requirements document is a waste of time as requirements change so quickly.
- The document is therefore always out of date.
- Methods such as XP use incremental requirements engineering and express requirements as 'user stories'
- This is practical for business systems but problematic for systems that require a lot of pre-delivery analysis (e.g. critical systems) or systems developed by several teams.

## Users of a Requirements Document



Chapter 3 Requirements engineering

31

## 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.

Chapter 3 Requirements engineering

32



## The structure of a requirements document

Chapter	Description
Preface (mở đầu)	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 (giới thiệu)	This should describe the need for the system. It should briefly describe the system's functions and explain how it will work with other systems. It should also describe how the system fits into the overall business or strategic objectives of the organization commissioning the software.
Glossary (thuật ngữ)	This should define the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.
User requirements definition (xác định yêu cầu người dùng)	Here, you describe the services provided for the user. The nonfunctional system requirements should also be described in this section. This description may use natural language, diagrams, or other notations that are understandable to customers. Product and process standards that must be followed should be specified.
System architecture (kiến trúc hệ thống)	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.

## Structure of a Requirements Document

Chapter	Description
System requirements specification (Đặc tả yêu cầu hệ thống)	This should describe the functional and nonfunctional requirements in more detail. If necessary, further detail may also be added to the nonfunctional requirements. Interfaces to other systems may be defined.
System models (mô hình hệ thống)	This might include graphical system models showing the relationships between the system components and the system and its environment. Examples of possible models are object models, data-flow models, or semantic data models.
System evolution (tiến hóa hệ thống)	This should describe the fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on. This section is useful for system designers as it may help them avoid design decisions that would constrain likely future changes to the system.
Appendices (phụ lục)	These should provide detailed, specific information that is related to the application being developed; for example, hardware and database descriptions. Hardware requirements define the minimal and optimal configurations for the system. Database requirements define the logical organization of the data used by the system and the relationships between data.
Index (chỉ mục)	Several indexes to the document may be included. As well as a normal alphabetic index, there may be an index of diagrams, an index of functions, and so on.

## Requirements Specification

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

## Ways of writing a System Requirements Specification

Notation	Description
<b>Natural language</b>	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
Design description languages	This approach uses a language like a programming language, but with more abstract features to specify the requirements by defining an operational model of the system. This approach is now rarely used although it can be useful for interface specifications.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system; UML use case and sequence diagrams are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification. They cannot check that it represents what they want and are reluctant to accept it as a system contract



## Requirements and Design

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



## Natural Language Specification

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



## 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
  - 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

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 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 system but is sometimes too rigid for writing business system requirements.



## Form-based Specifications

- 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.



## 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 systems 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.

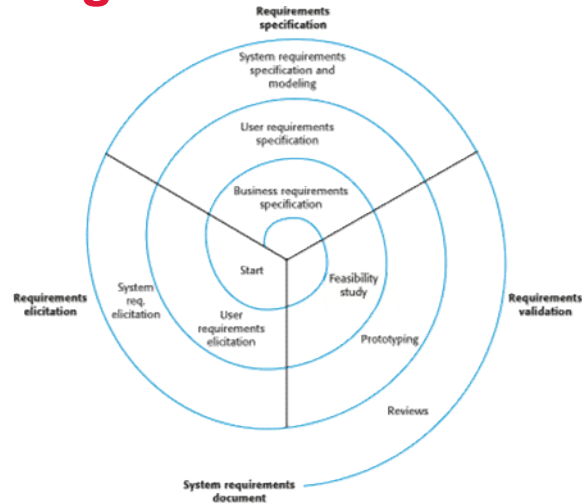
## Tabular Specification of Computation for an Insulin Pump

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

## 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 (khơi gợi yêu cầu);
  - Requirements analysis (phân tích yêu cầu);
  - Requirements validation (phê chuẩn yêu cầu);
  - Requirements management (quản lý yêu cầu).
- In practice, RE is an iterative (lặp) activity in which these processes are interleaved (xen kẽ).

## A Spiral View of the Requirements Engineering Process



Chapter 3 Requirements engineering

47

## 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*.

Chapter 3 Requirements engineering

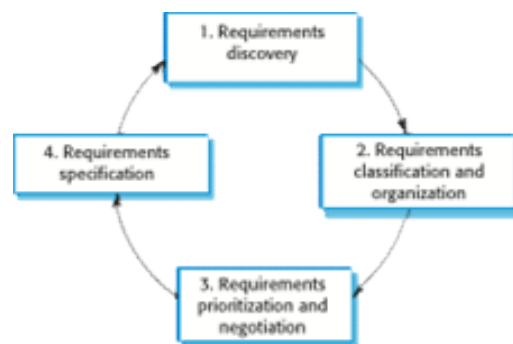
48



## Requirements Elicitation and Analysis

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

## 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.


## 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.
- Organisational and political factors may influence the system requirements.
- The requirements change during the analysis process. New stakeholders may emerge and the business environment change.



# Requirements Engineering

Chapter 3 Requirements engineering 53



## 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.

Chapter 3 Requirements engineering 54



## Stakeholders in the MHC-PMS

- 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 MHC-PMS

- 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.

## 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.

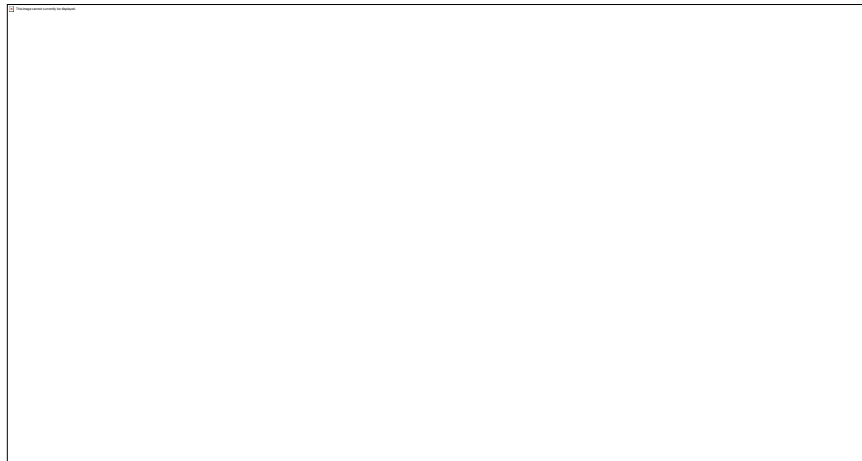
## Interviews in Practice

- 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.
- 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.

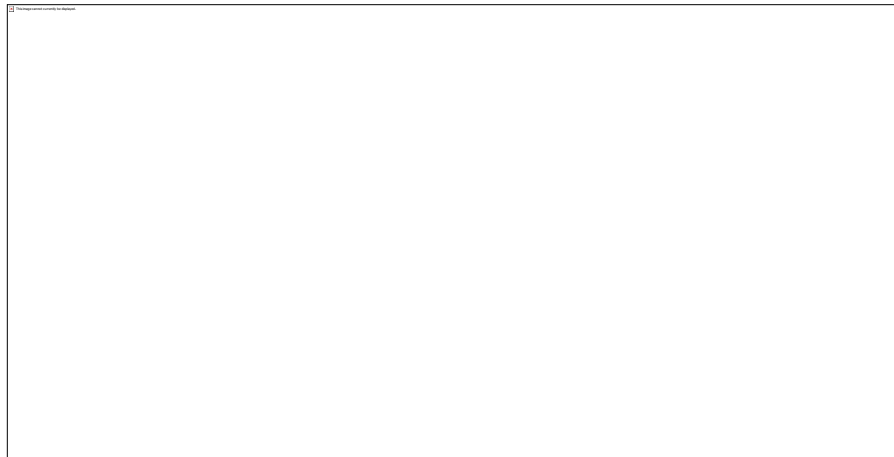
## Scenarios

- Scenarios are real-life examples of how a system can be used.
- They 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.

## Scenario for collecting medical history in MHC-PMS



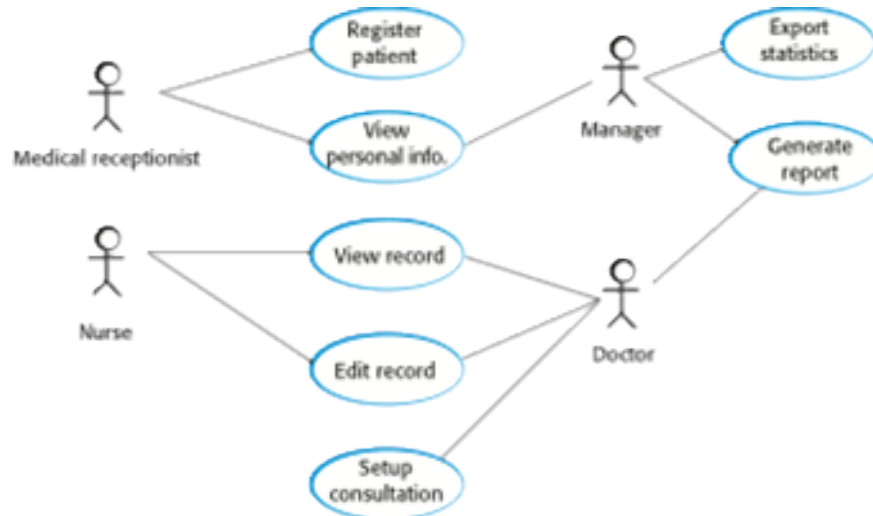
## Scenario for collecting medical history in MHC-PMS



## Use cases

- Use-cases are a scenario based technique in the UML which 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
- 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 MHC-PMS



Chapter 3 Requirements engineering

63

## Requirements Validation

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

Chapter 3 Requirements engineering

64





## Requirements Checking

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



## Requirements 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 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.

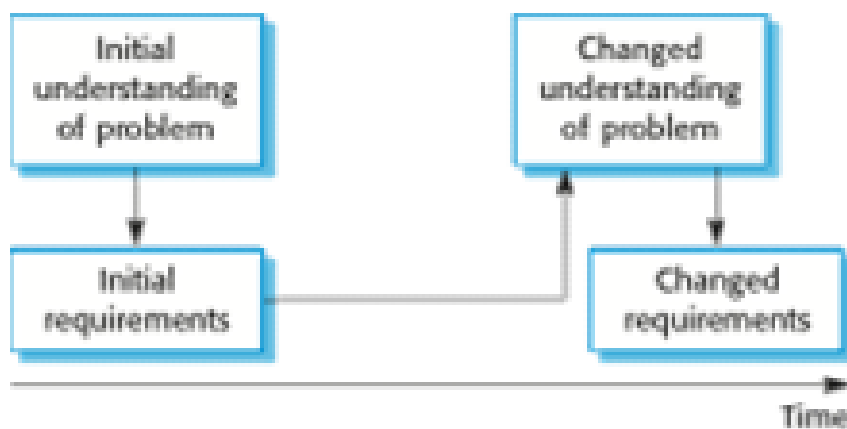
## Changing Requirements

- 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 if the system is to meet its goals.

## Changing Requirements

- 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 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* This is the set of activities that assess the impact and cost of changes. I discuss this process in more detail in the following section.
  - *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.

Chapter 3 Requirements engineering

73

## Requirements Change Management

- Deciding if a requirements change should be accepted
  - *Problem analysis and change specification*
    - During this stage, the problem or the change proposal is analyzed to check that it is valid. This analysis is fed back to the change requestor who may respond with a more specific requirements change proposal, or decide to withdraw the request.
  - *Change analysis and costing*
    - The effect of the proposed change is assessed using traceability information and general knowledge of the system requirements. Once this analysis is completed, a decision is made whether or not to proceed with the requirements change.
  - *Change implementation*
    - The requirements document and, where necessary, the system design and implementation, are modified. Ideally, the document should be organized so that changes can be easily implemented.

Chapter 3 Requirements engineering

74

# Requirements Change Management

