Department Of Computer Science

Lecture 04

Requirements Engineering

Objectives

- Requirement (Definition and Introduction)
- **Requirement Engineering Process**
  - Requirements elicitation and analysis
  - Requirements specification
  - The software requirements document **(SRS)**
- User, System and domain requirements.
- Types: **Functional** and **Non-Functional** requirements
  - Requirement Characteristics
- Requirement Gathering Techniques
- Requirements Validation
- Requirements Management

What is a Requirement?

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

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**."

Agile Methods and Requirements

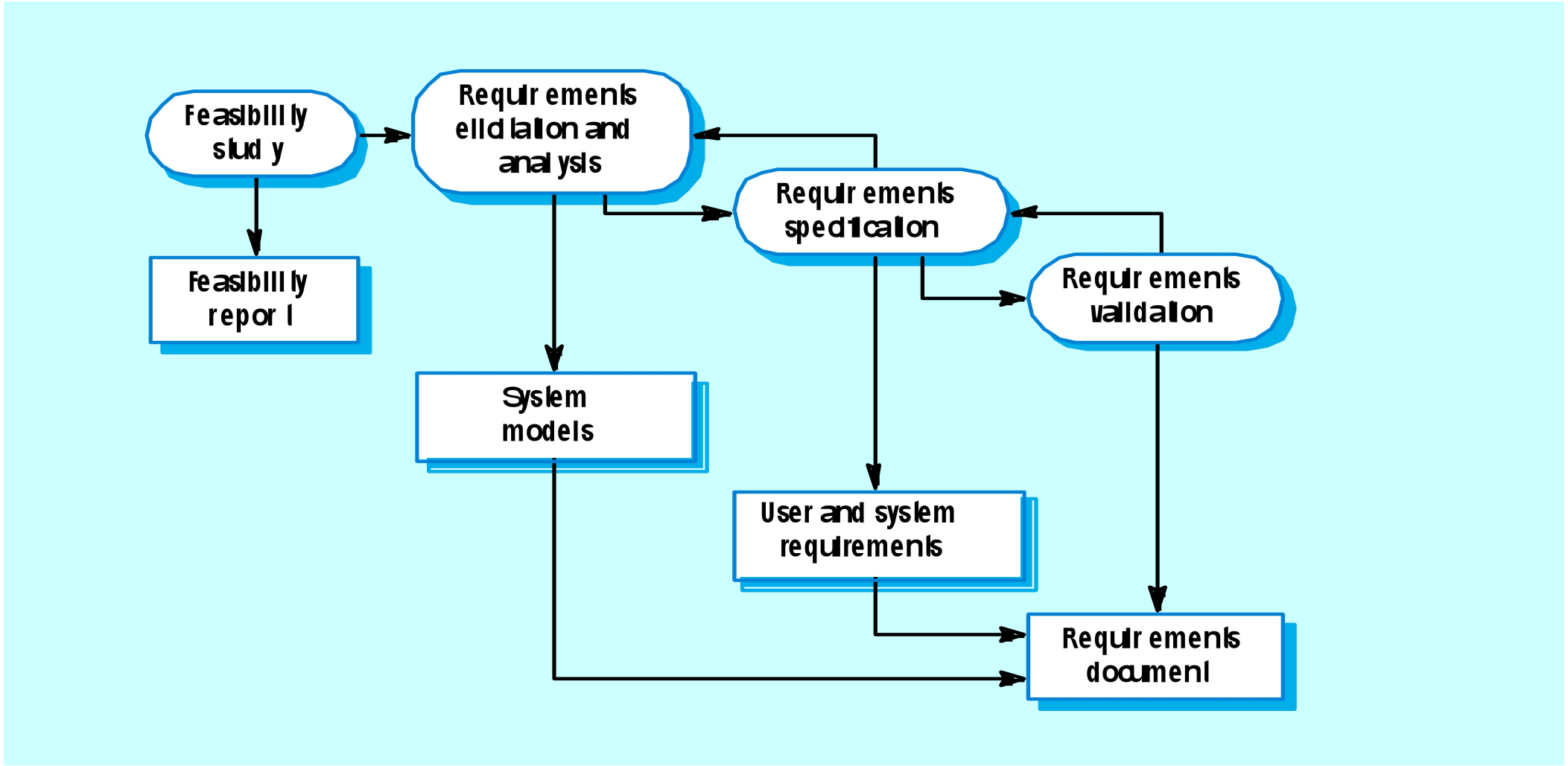
- **Many agile methods** argue that producing a requirements document is a waste of time as requirements change so quickly.
- **The document** is therefore always out of date.
- **Methods such as XP use** incremental requirements engineering and express requirements as ‘user stories’
- **This is practical 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.

- **Requirement Engineering**

Requirements Engineering

- The requirements for a system are the descriptions of **what the system should do**
- The process of establishing the **services** that the customer requires from a system and the **constraints** under which it operates and is developed.
- The process of finding out, analyzing, documenting and checking these services and constraints is called **requirements engineering (RE)**

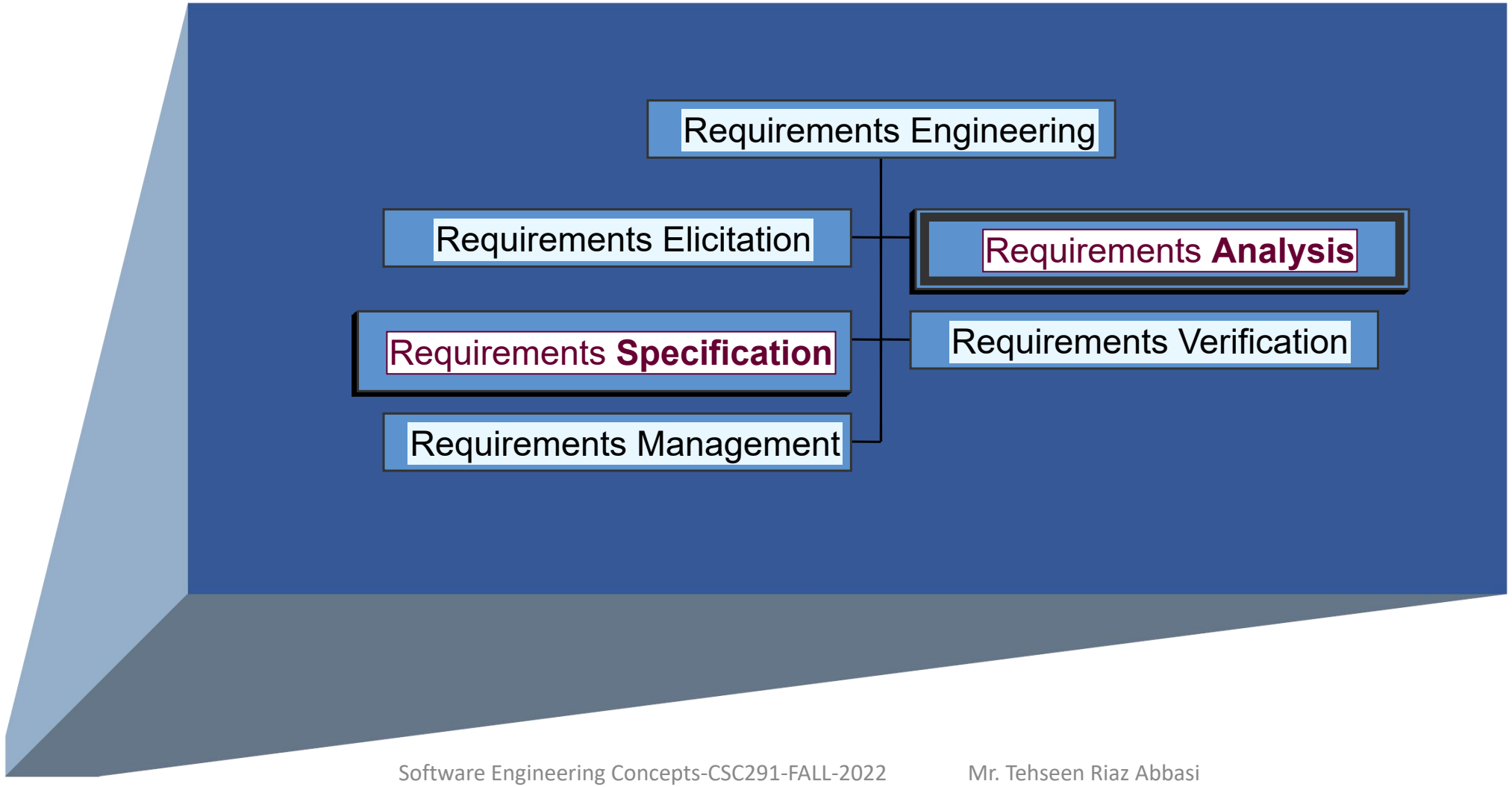
The Requirements Engineering Process



Requirements Engineering Process

- **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 generic activities common to all processes
  - Requirements elicitation;
  - Requirements analysis;
  - Requirements Specification;
  - Requirements validation;
  - Requirements management.

Requirements Engineering



Steps in Requirement Phase

The requirements part of a project can be divided into several stages:

- Analysis to establish the system's services, constraints, and goals by consultation with client, customers, and users.
- Modelling to organize the requirements in a systematic and comprehensible manner.
- Define, record, and communicate the requirements.

Requirements Elicitation and Analysis

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

Requirements Elicitation 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.**

The Requirements Elicitation and Analysis Process

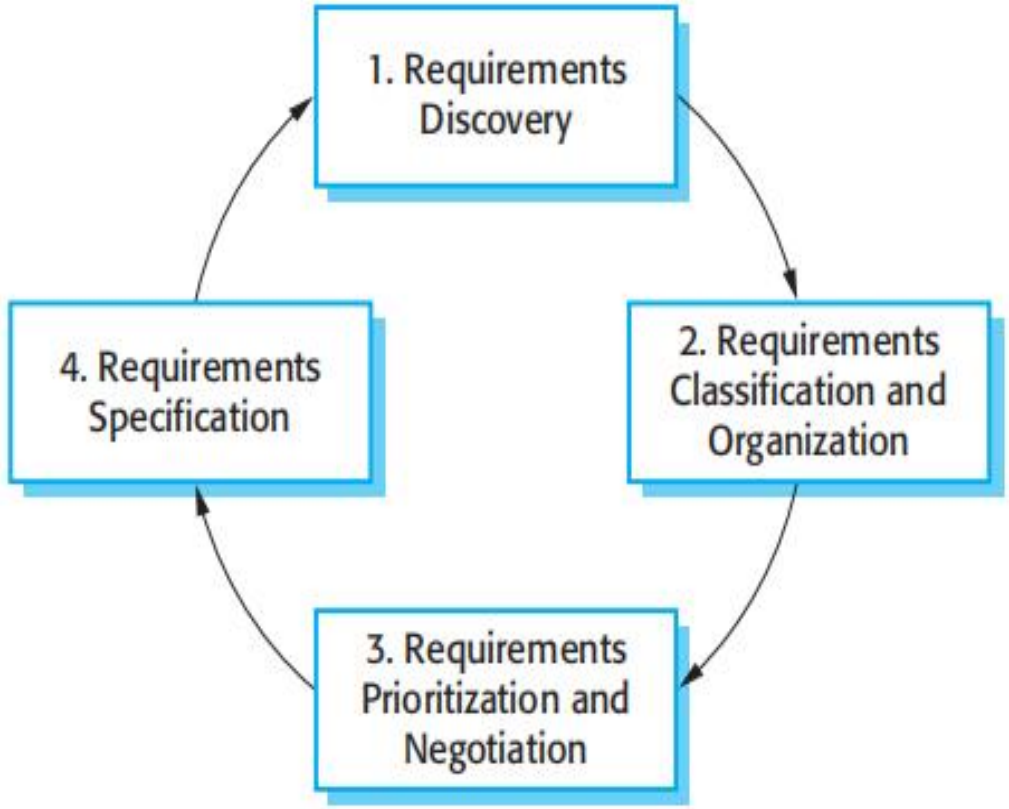


Figure 4.13 The requirements elicitation and analysis process

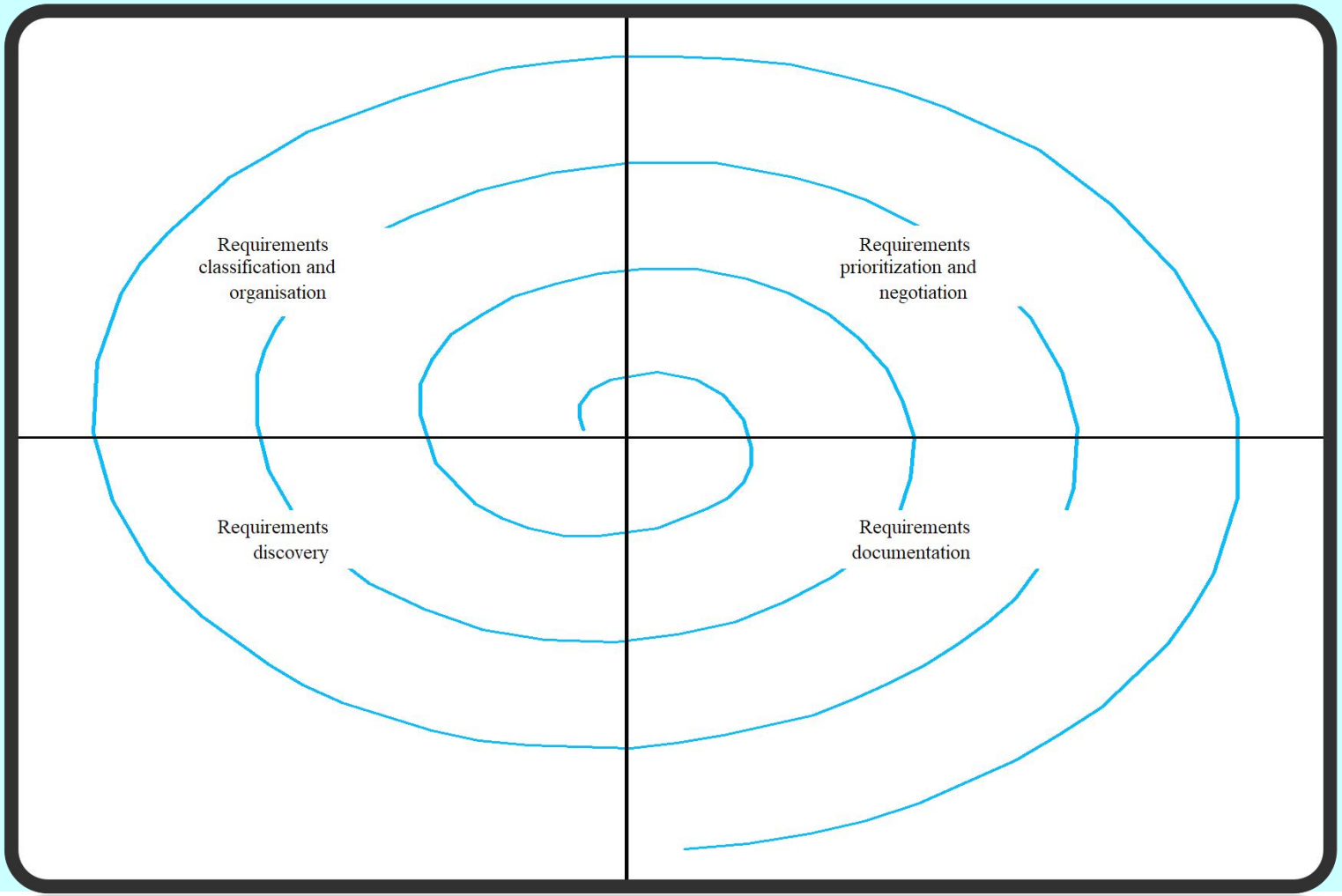
Requirements Elicitation and analysis

- **Requirements discovery:** This is the process of interacting with stakeholders of the system to discover their requirements. (Using requirements gathering techniques)
- **Requirements classification and organization:** This activity takes the unstructured collection of requirements, groups related requirements, and organizes them into coherent clusters.
- **Requirements prioritization and negotiation:** Inevitably, when multiple stake holders are involved, requirements will conflict. This activity is concerned with prioritizing requirements and finding and resolving requirements conflicts through negotiation.
- **Requirements specification.** The requirements are documented and input into the next round of the spiral. Formal or informal requirements documents may be produced.

Problems of Requirements Analysis

- **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.

The Requirements Spiral



A spiral view of the requirements engineering process

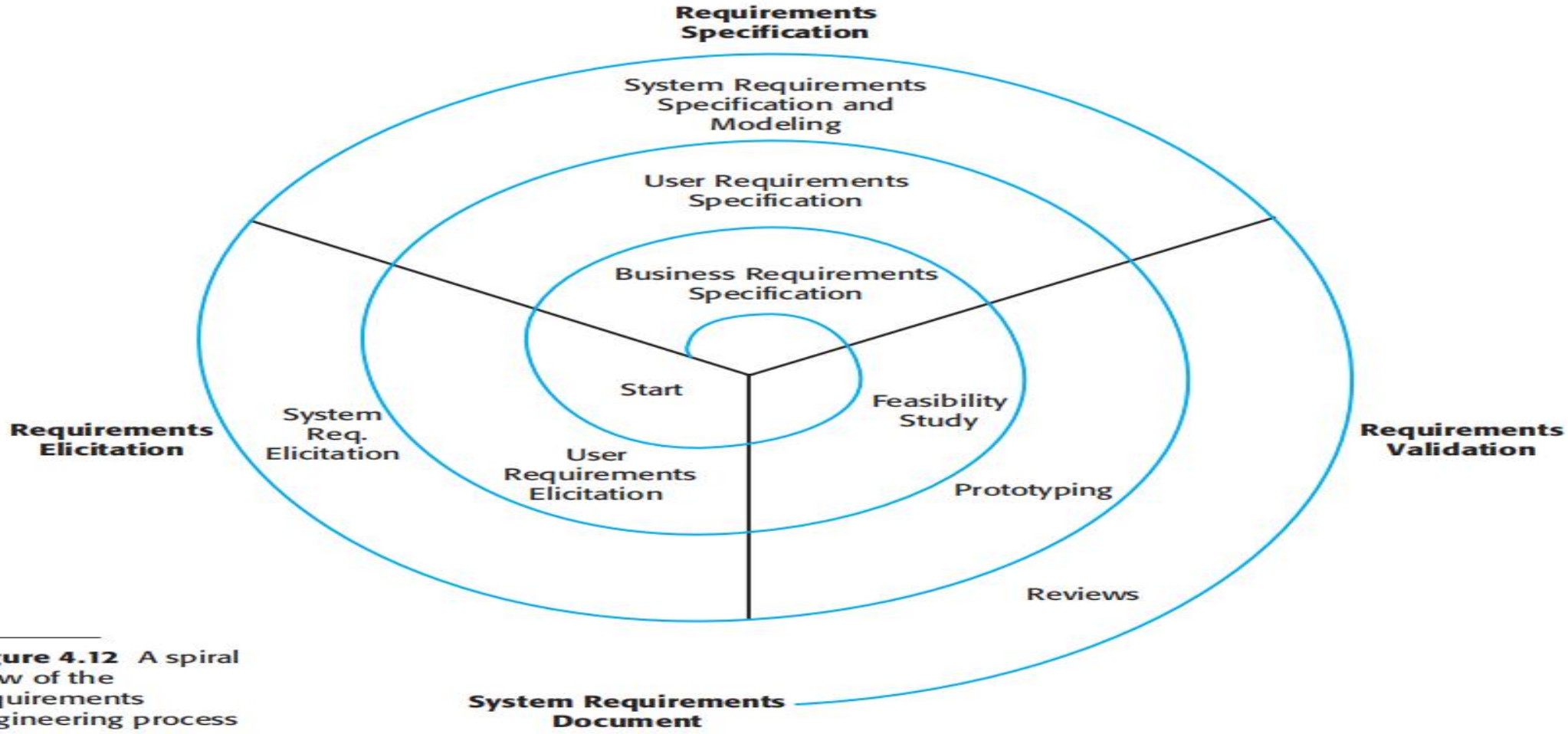


Figure 4.12 A spiral view of the requirements engineering process

Process activities

- **Requirements discovery**
  - Interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage.
- **Requirements classification and organization**
  - Groups related requirements and organises 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 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.

- **Requirements Specification**

Requirements Specification

- **Requirement Specification: The process of writing don 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.
- **From user and System Requirements, the outcome is a requirements document, which may be part of the**
- system development contract.
- **The requirements may be part of a contract for the system development**
  - It is therefore important that these are as complete as possible.

Requirements Specification

User Requirement Definition

1. The MHC-PMS 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 automatically 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., 10 mg, 20 mg) separate reports shall be created for each dose unit.

1.5 Access to all cost reports shall be restricted to authorized users listed on a management access control list.

Figure 4.1 User and system requirements

Software Requirement Specification

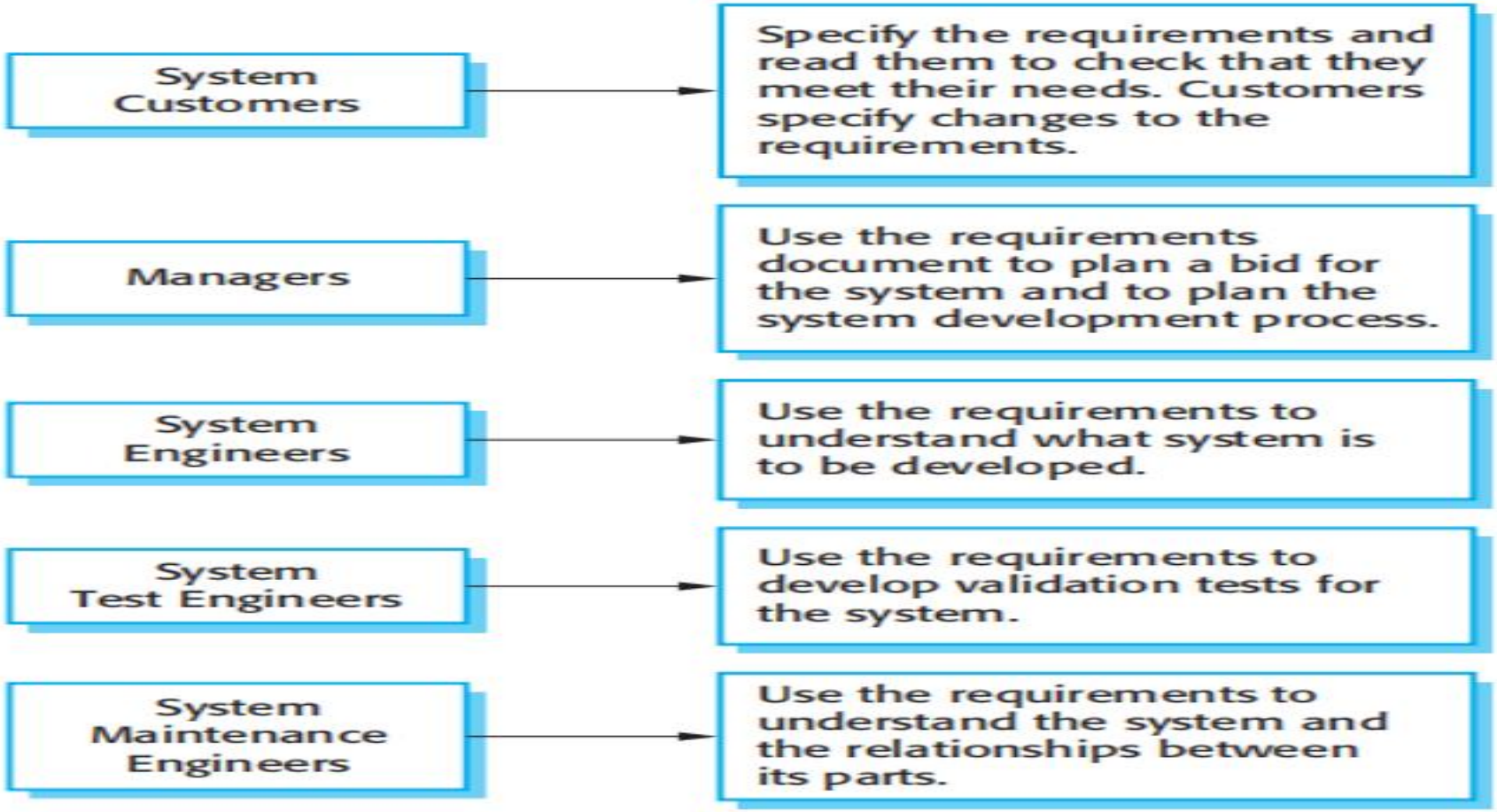
- **Requirements specification** Requirements specification is the process of writing down the user and system requirements in a requirements document.
- A software requirements specification (SRS) is a complete description of the behavior of the system to be developed
- A document that clearly and precisely describes, each of the essential requirements of the software and the external interfaces.
  - (functions, performance, design constraint, and quality attributes)

- Each requirement is defined in such a way that its achievement is capable of being *objectively verified* by a prescribed method; for example inspection, demonstration, analysis, or test.

The Software Requirements Document

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

Users of a Requirements Document



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.

- Discuss SRS document Template

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 **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 will 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 will 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.)*

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.

- **Requirements Context:**

System, User and Domain Requirements

Types of Requirement (Context)

- **User Requirements**
  - Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.
- **System Requirements**
  - A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

User and System Requirements

User Requirement Definition

1. The MHC-PMS 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 automatically 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., 10 mg, 20 mg) separate reports shall be created for each dose unit.

1.5 Access to all cost reports shall be restricted to authorized users listed on a management access control list.

Figure 4.1 User and system requirements

Readers of Different types of Requirements Specification

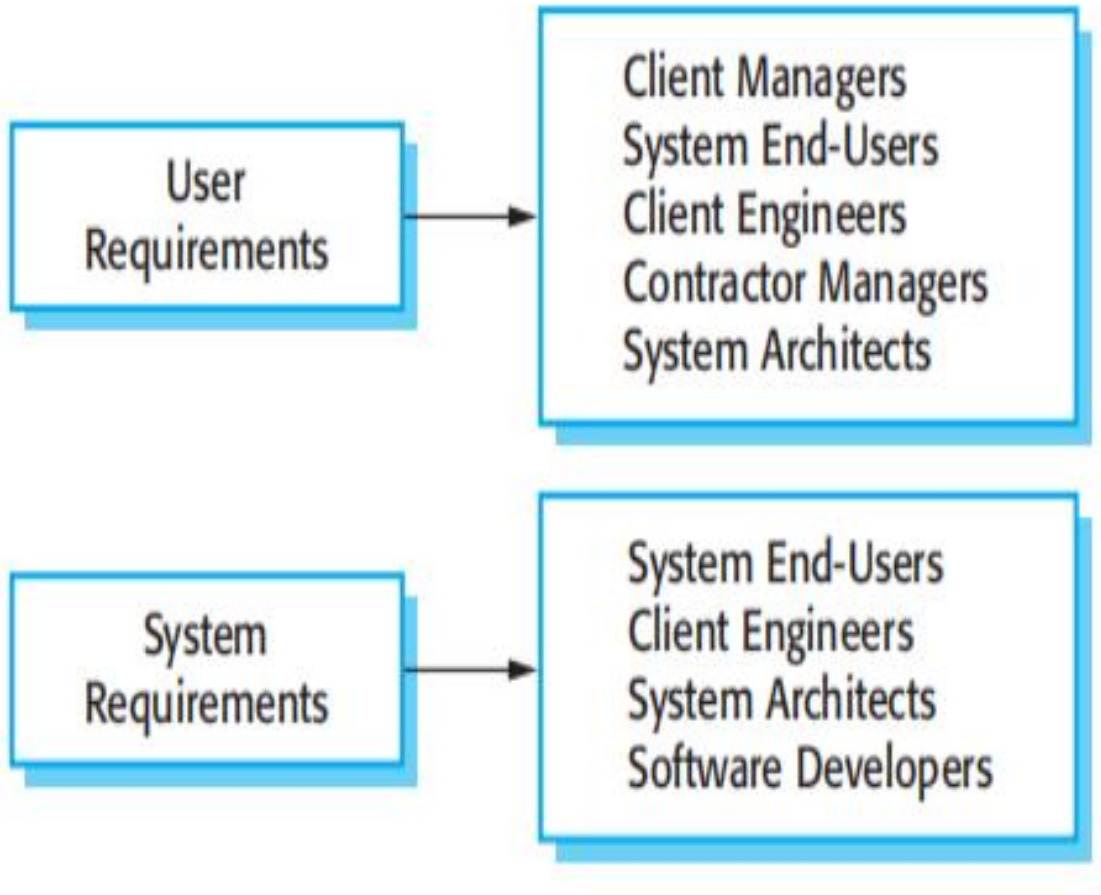


Figure 4.2 Readers of different types of requirements specification

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 will be computed as:
  - Dtrain = Dcontrol + Dgradient
  - where Dgradient is 9.81ms2 * compensated gradient/alpha and where the values of 9.81ms2 /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.**

- **Types of Requirements:**

Functional Requirements and Non-Functional Requirement

Types of Requirements: Functional and Non-Functional Requirements

- **Functional requirements**
  - Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
  - **May state what the system should not do.**
- **Non-Functional requirements**
  - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
  - **Often apply to the system as a whole rather than individual features or services.**
  - **In other words,** a non-functional requirement will describe how a system should behave and what limits there are on its functionality.

Functional Requirements

- **What the system should do**
- **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 for the MHC-PMS

- A **user will be able to search** the appointments lists for all clinics.
- **The system will generate each day**, for each clinic, a list of patients who are expected to attend appointments that day.
- **Each staff member** using the system will be uniquely identified by his or her 8-digit employee number.

Non-functional requirements

- **These define system properties and constraints**
  - e.g. reliability, response time and storage requirements.
- **Constraints on the system implementation**
  - e.g. I/O device capability, data representations, etc.
- **Constrain characteristics** of the system as a whole
  - such as performance, security, or availability
- **Non-functional requirements** may be more critical than functional requirements. If these are not met, the system may be useless.
  - **For example**, if an aircraft system does not meet its reliability requirements, it will not be certified as safe for operation
  - if an embedded control system fails to meet its performance requirements, the control functions will not operate correctly

Non-functional requirements implementation

- Non-functional requirements may affect the overall architecture of a system rather than the individual components.
  - **For example**, to ensure that performance requirements are met, you may have to organize the system to minimize communications between components.
- A single non-functional requirement, such as a security requirement, may generate a number of related functional requirements that define system services that are required.
  - It may also generate requirements that restrict existing requirements.

Non-functional Requirements

Non-functional Requirements

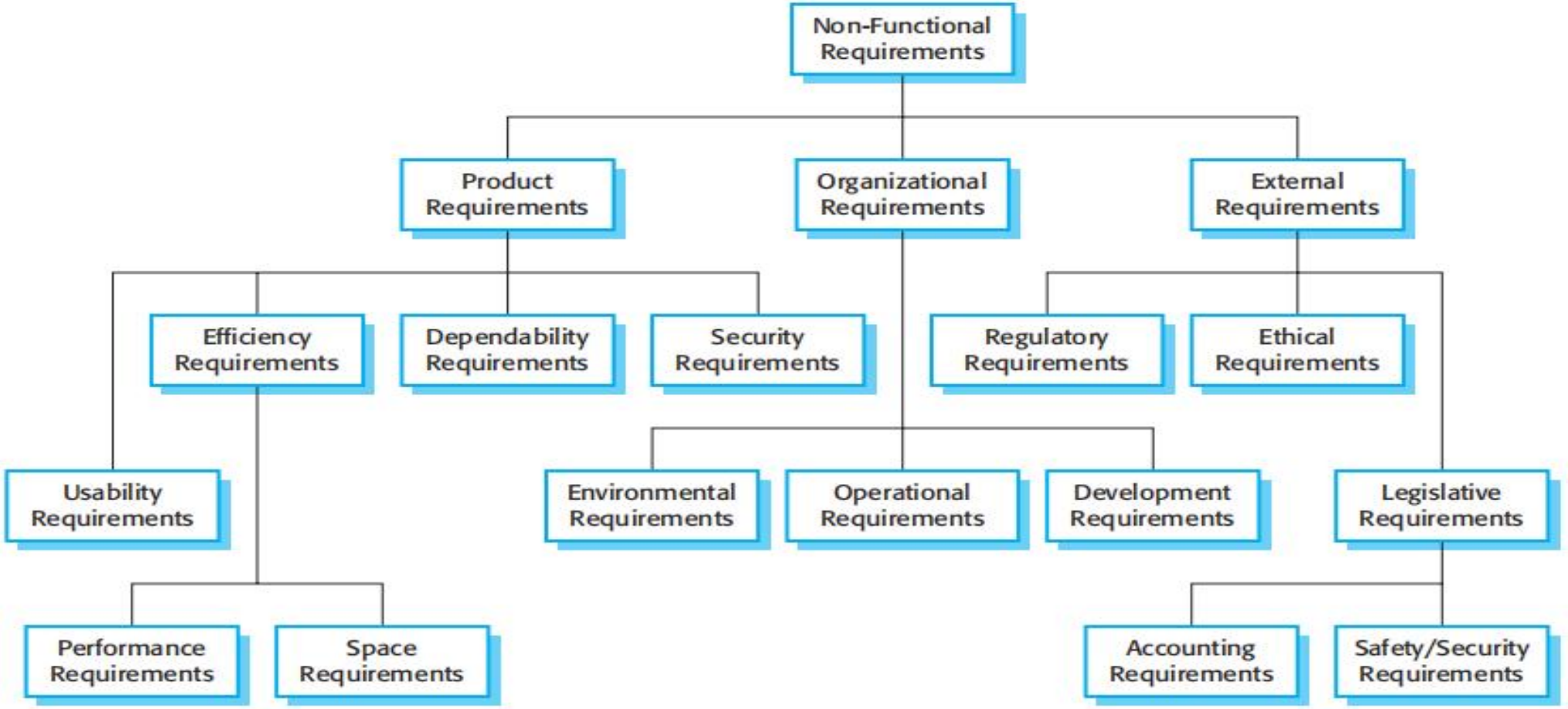
The FURPS Model (Functionality, Usability, Reliability, Performance, and Supportability)

Typical non-functional requirements include:

- Performance– for example: response time, throughput, utilization, static volumetric
- Scalability
- Capacity
- Availability
- Reliability
- Recoverability
- Maintainability
- Serviceability
- Security
- Regulatory
- Manageability
- Environmental
- Data Integrity
- Usability
- Interoperability

Visit: https://requirementsquest.com/nonfunctional-requirement-examples/

Types of Non-Functional Requirement



Non-Functional Requirement 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.

Examples of non-functional requirements in the MHC-PMS

Product requirement
The MHC-PMS will be available to all clinics during normal working hours (Mon–Fri, 0830–17.30). Downtime within normal working hours will not exceed five seconds in any one day.

Organizational requirement
Users of the MHC-PMS system will authenticate themselves using their health authority identity card.

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

NFR- Goals and Requirements

- **A common problem with non-functional requirements** is that users or customers often propose these requirements as general goals
- **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

- **How a manager might express usability requirements as GOAL**
  - The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized. (Goal)

- **How the goal could be expressed as a 'testable' nonfunctional requirement**
  - Medical staff will 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 will not exceed two per hour of system use. (Testable non-functional requirement)

Metrics for Specifying Non-Functional 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

Functional and Non-Functional Requirements

Non Functional vs. Functional Requirements

Here, are key differences between Functional and Nonfunctional requirements in Software Engineering:

Parameters	Functional Requirement	Non-Functional Requirement
What it is	Verb	Attributes
Requirement	It is mandatory	It is non-mandatory
Capturing type	It is captured in use case.	It is captured as a quality attribute.
End result	Product feature	Product properties
Capturing	Easy to capture	Hard to capture
Objective	Helps you verify the functionality of the software.	Helps you to verify the performance of the software.
Area of focus	Focus on user requirement	Concentrates on the user's expectation.
Documentation	Describe what the product does	Describes how the product works
Type of Testing	Functional Testing like System, Integration, End to End, API testing, etc.	Non-Functional Testing like Performance, Stress, Usability, Security testing, etc.
Test Execution	Test Execution is done before non-functional testing.	After the functional testing
Product Info	Product Features	Product Properties

- **Requirements Characteristics**

Requirements Imprecision

- **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.

Requirements Characteristic

- **Good requirements should have the following characteristics:**
- Unambiguous
- Testable (verifiable)
- Clear (concise, terse, simple, precise)
- Correct
- Understandable
- Feasible (realistic, possible)
- Independent
- Atomic
- Necessary
- Implementation-free (abstract)

Good and Bad Requirements

- **R1:** Only authorized persons are allowed to enter the building.
- **R2**: The doorbell button will be blue, round, and with a diameter of 20 mm.
  - R1 is a goal,
  - R2 is a detailed specification
  - R2 is in fact three different requirements.

Good and Bad Requirements - Example 2

- **R4**: The system will play the sound ’alarm’ when the door is opened.

- **R4:** The system will play the sound ’C:\alarm.wav’ once, at the highest volume setting in the main speaker for the building when the door opens.

Good and Bad Requirements - Example 3

- **R5:** The system will respond quickly to user interaction.

- **R5:** The system will respond to user interaction within maximum 2 seconds, minimum 0.2 seconds, average 1 second (with a CPU load of 0.5).

Good and Bad Requirements - Example 4

- **R6:** There are four classes of users in the system: Maintainer, Administrator, Normal User, Guest.
- **R7:** A Guest User is able to enter areas X1, X2, and X3.
- **R8:** A Normal User is able to enter all areas in the building.
- **R9:** An Administrator is a Normal User who also have the right to add/modify/delete users in the system.
- **R10:** A Maintainer is a Guest who is also able to enter area X99 and perform backups, view system logs, and add/modify/delete users.

Example 4 cont…..

•**R6:** Users and their access rights are listed in Table 1

User Class	Area X1	…….	Area X99	Add/modify/ delete users	Backups	View Logs
Guest	Yes		No	No	No	No
Normal user	Yes		Yes	No	No	No
Administrator	Yes		Yes	Yes	No	Yes
Maintainer	NO		Yes	Yes	Yes	Yes

Good and Bad Requirements - Example 5

- **R11:** When a user enters a correct code, the door will open.

- **R11:** When a user enters the code associated with him/her on keypad K2 the door D2 will unlock and remain unlocked for 30 seconds.

Good and Bad Requirements - Example 6

- **R12:** When a user has entered the building, this will be registered in the entry log and the user's computer will be booted.
- **R12:** When a user has entered the building, this will be registered in the entry log.
- **R13:** When a user has entered the building, the user's computer will be booted.
- **R12:** The system will register in the entry log when a user has entered the building.

Requirements Gathering Techniques

Requirements Discovery - Elicitation

- **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.
  - Sources of information may include documentation, system stakeholders, and specifications of similar systems
- **Systems normally have a range of stakeholders.**

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.

Techniques of Eliciting Requirements

- Analysts can employ several techniques to elicit the requirements from the customer.
  - **Survey/Questionnaire**
  - **Focus groups** (requirements workshops) and creating requirements lists.
  - **Naturalistic observation**
  - **Document Analysis**
  - **Interviews**
  - **Ethnography**
  - **Prototyping**
  - **Brainstorming**
  - **Scenarios**
  - **Use cases**
    - Combination of these methods will be good choice.

Questionnaires

- **Questionnaires:** Series of questions designed to elicit specific information from us. The questions may require different kinds of answers: some require a simple Yes/No, others ask us to choose from a set of pre-supplied answers.

- **SurveyMonkey**
  - Popular solution to let you design and distribute questionnaires
  - Runs ‘in the cloud’
- **Lime Survey**
  - A free online tool that can be installed directly onto the researcher’s system, thus avoiding storage of data in the cloud (better control of confidential data)

Focus groups and workshops

- **Focus groups and workshops:** Interviews tend to be one on one, and elicit only one person's perspective. It can be very revealing to get a group of stakeholders together to discuss issues and requirements.

- **Focus Group: For the focus group following are the key things**:
  1. Discussion is focused around a topic.
  2. Facilitator already knows what he wanted to ask from each participants and develop a discussion guide accordingly.
  3. Facilitator develops discussion guide based on experience of each participants with the product.
  4. Based on discussion guided by discussion guide product requirements are identified.

Naturalistic Observation

- **Naturalistic Observation**: It can be very difficult for humans to explain what they do or to even describe accurately how they achieve a task.
  - Spend time with stakeholders in their day-to-day tasks, observing work as it happens
  - Gain insights into stakeholders' tasks
  - Good for understanding the nature and context of the tasks
  - But it requires time and commitment from a member of the design team, and it can result in a huge amount of data
  - **Ethnography is one form** : entire class devoted to this.
- **Directly Observing Users**
  - Serves as a good method to supplement interviews
  - Often difficult to obtain unbiased data
  - People often work differently when being observed
  - Be cognizant of normal and abnormal conditions, e.g. entering an order vs. the end of quarter sales report

Document Analysis

- **Studying documentation:** Procedures and rules are often written down in a manual and these are a good source of data about the steps involved in an activity and any regulations governing a task.

- **Four types of useful documents**
  - **Written work procedures**
    - Describes how a job is performed
    - Includes data and information used and created in the process of performing the job or task
  - **Business form**
    - Explicitly indicate **data flow** in or out of a system
  - **Report**
    - Enables the analyst to work backwards from the report to the data that generated it
  - **Description of current information system**

Interviewing

- **Interviews:** Interviews involve asking someone a set of questions. Often interviews are face-to-face, but they don't have to.
- Interviews are good for getting an overall understanding of what stakeholders do, how they might interact with the new system, and the difficulties that they face with current systems.
- 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.

Ethnography

Ethnography is an observational technique that can be used to understand operational processes and help derive support requirements for these processes. An analyst immerses himself or herself in the working environment where the system will be used.

Ethnography is an observational technique that can be used to understand operational processes and help derive requirements for these processes. It helps discover the implicit system requirements that reflect the actual way that people work rather than formal processes defined by the organization.

Ethnography

Particularly useful for discovering the following two types of requirements:

Requirements that are derived from the way in which people actually work, rather than the way in which process definitions say they ought to work.

For example, air traffic controllers may switch off a conflict alert system that detects aircraft with intersecting flight paths, even though normal control procedures specify that it should be used. They deliberately put the aircraft on conflicting paths for a short time to help manage the airspace. Their control strategy is designed to ensure that these aircraft are moved apart before problems occur and they find that the conflict alert alarm distracts them from their work.

Requirements that are derived from cooperation and awareness of other people's activities.

For example, air traffic controllers may use an awareness of other controllers' work to predict the number of aircrafts that will be entering their control sector. They then modify their control strategies depending on that predicted workload. Therefore, an automated ATC system should allow controllers in a sector to have some visibility of the work in adjacent sectors.

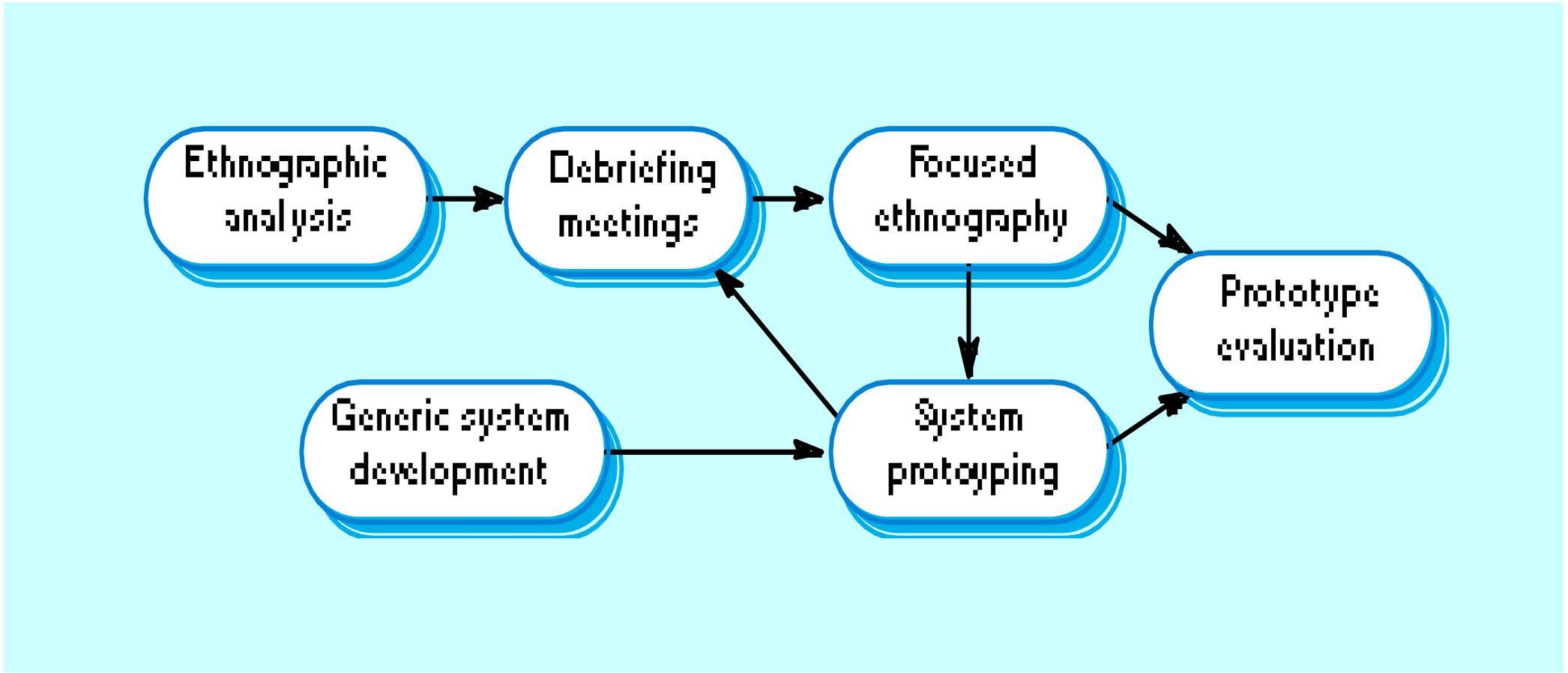
Ethnography

- **A social scientists spends a** considerable time observing and analyzing how people actually work.
- **People do not 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.

Focused Ethnography

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

Ethnography and Prototyping



Scope of Ethnography

- **Requirements that are derived** from the way that people actually work rather than the way I which process definitions suggest that they ought to work.
- **Requirements that are derived** from cooperation and awareness of other people's activities.

Prototyping

- Repetitive process
- Elementary version of system is built
- Replaces or augments SDLC
- **Goal:** to develop concrete specifications for ultimate system
- Quickly converts requirements to working version of system
- Once the user sees requirements converted to system, will ask for modifications or will generate additional requests
- **Is prototyping useful in any of these cases?**
  - User requests are not clear
  - Few users are involved in the system
  - Designs are complex and require concrete form
  - History of communication problems between analysts and users
  - Tools are readily available to build prototype

Brainstorming

- Brainstorming is a situation where a group of people meet to generate new ideas and solutions around a specific domain of interest by removing inhibitions. People are able to think more freely, and they suggest as many spontaneous new ideas as possible. All the ideas are noted down without criticism and after the brainstorming session the ideas are evaluated
- **Individual Brainstorming**
- **Group Brainstorming**

Task descriptions

- **Scenarios**
  - An informal narrative story of users
  - Natural way to explain
  - Scenarios can be particularly useful for adding detail to an outline requirements description.
- **Use cases**
  - Show interaction with a system
  - Show detailed understanding of the interaction
- **Essential use cases**
  - Improvement of use cases
- Often used in combination

- **Scenarios And Use Cases**

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

INITIAL ASSUMPTION:
The patient has seen a medical receptionist who has created a record in the system and collected the patient's personal information (name, address, age, etc.). A nurse is logged on to the system and is collecting medical history.

NORMAL:
The nurse searches for the patient by family name. If there is more than one patient with the same surname, the given name (first name in English) and date of birth are used to identify the patient.

The nurse chooses the menu option to add medical history.

The nurse then follows a series of prompts from the system to enter information about consultations elsewhere on mental health problems (free text input), existing medical conditions (nurse selects conditions from menu), medication currently taken (selected from menu), allergies (free text), and home life (form).

WHAT CAN GO WRONG:
The patient's record does not exist or cannot be found. The nurse should create a new record and record personal information.

Patient conditions or medication are not entered in the menu. The nurse should choose the 'other' option and enter free text describing the condition/medication.

Patient cannot/will not provide information on medical history. The nurse should enter free text recording the patient's inability/unwillingness to provide information. The system should print the standard exclusion form stating that the lack of information may mean that treatment will be limited or delayed. This should be signed and handed to the patient.

OTHER ACTIVITIES:
Record may be consulted but not edited by other staff while information is being entered.

SYSTEM STATE ON COMPLETION:
User is logged on. The patient record including medical history is entered in the database, a record is added to the system log showing the start and end time of the session and the nurse involved.

Figure 4.14 Scenario for collecting medical history in MHC-PMS

Scenario-based elicitation involves working with stakeholders to identify scenarios and to capture details to be included in these scenarios. Scenarios may be written as text, supplemented by diagrams, screen shots, etc. Alternatively, a more structured approach such as event scenarios or use cases may be used.

- **Use Cases**

Use cases

- **A use case identifies the actors involved in an interaction and names the type of interaction. Use cases are documented using a high-level use case diagram.**
- **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.
- Sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system.

Use Cases

- A use case is a collection of possible sequences of interactions between the system under discussion and its Users (or Actors), relating to a particular goal. The collection of Use Cases should define all system behaviour relevant to the actors to assure them that their goals will be carried out properly.
- There are many methods of defining how to pick or create a use case. For example, using a goal oriented Structuring Methodology presented by Alistair Cockburn of Humans and Technology. Examining all the Actor's goals that the system satisfies yields the functional requirements. Goals summarize system function in understandable verifiable terms of use that users, executives and developers can appreciate and leave little open to interpretation.

Use Cases (Cont.)

Use Cases:

- Hold Functional Requirements in an easy to read, easy to track text format.
- Represents the goal of an interaction between an actor and the system. The goal represents a meaningful and measurable objective for the actor.
- Records a set of paths (scenarios) that traverse an actor from a trigger event (start of the use case) to the goal (success scenarios).
- Records a set of scenarios that traverse an actor from a trigger event toward a goal but fall short of the goal (failure scenarios).
- Are multi-level: one use case can use/extent the functionality of another.

Use Cases Do Not…

- Specify user interface design. They specify the intent, not the action Detail
- Specify implementation detail (unless it is of particular importance to the actor to be assured that the goal is properly met)

Modelling Scenarios as Use Cases

- **Models**

  Scenarios are useful in discussing a proposed system with client, but requirements need to be made more precise before a system is fully understood.

- **This is the purpose of requirements modelling.**

- **A use case provides such a model.**
- **use case describes "who" can do "what" with the system in question**

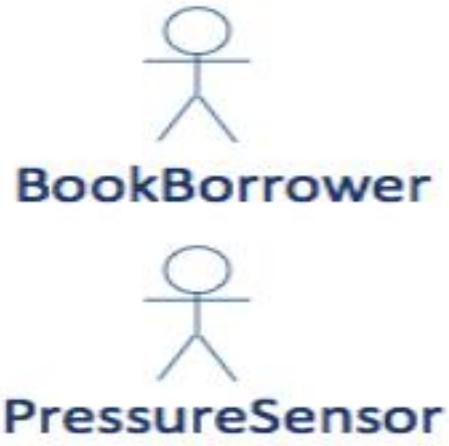
Use Case Diagrams

Use case diagrams

A use case diagram shows the relationships between actors and and their interactions with a system.

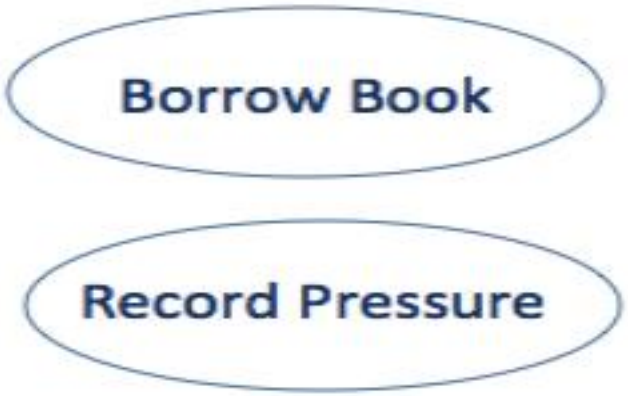
It does not show the logic of those interactions.

Actors and Use Case Diagrams



- An **actor** is a user of a system in a particular **role**.

  An actor can be human or an external system.



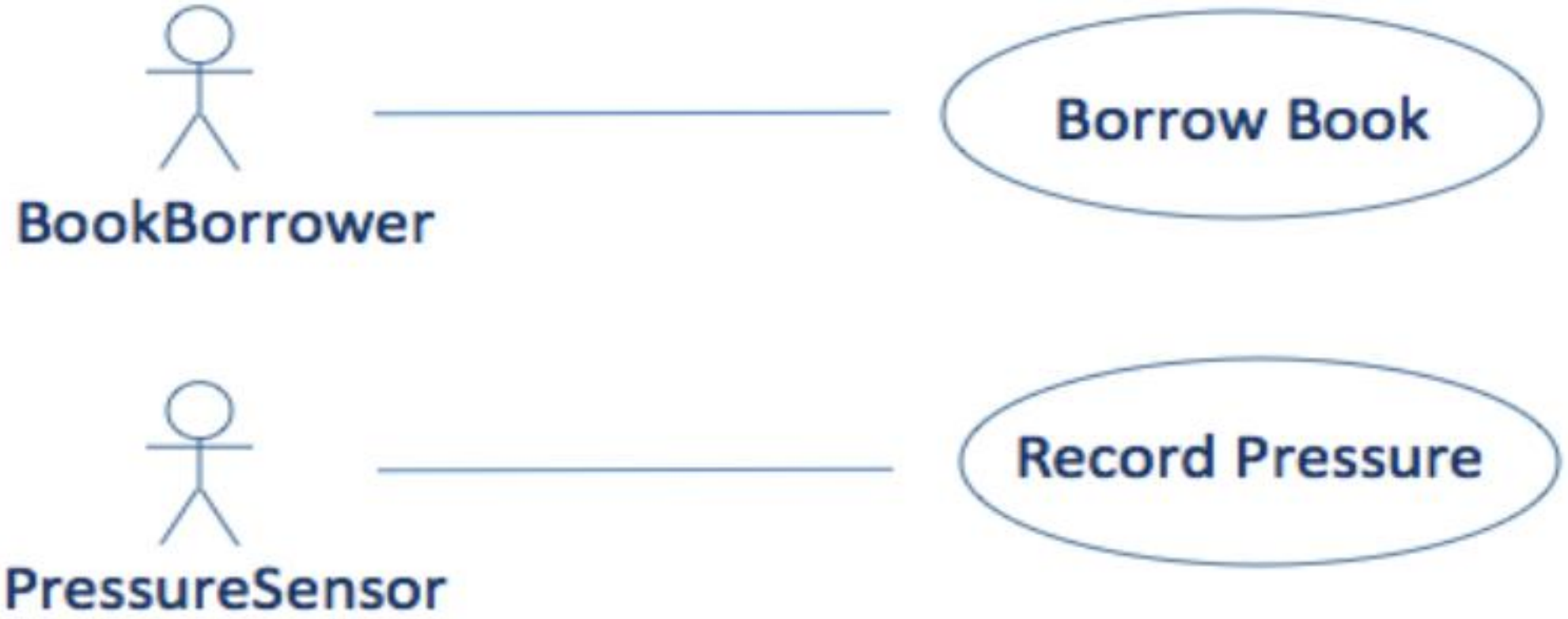
- A **use case** is a a task that an actor needs to perform with the help of the system.

Use Cases and Actors

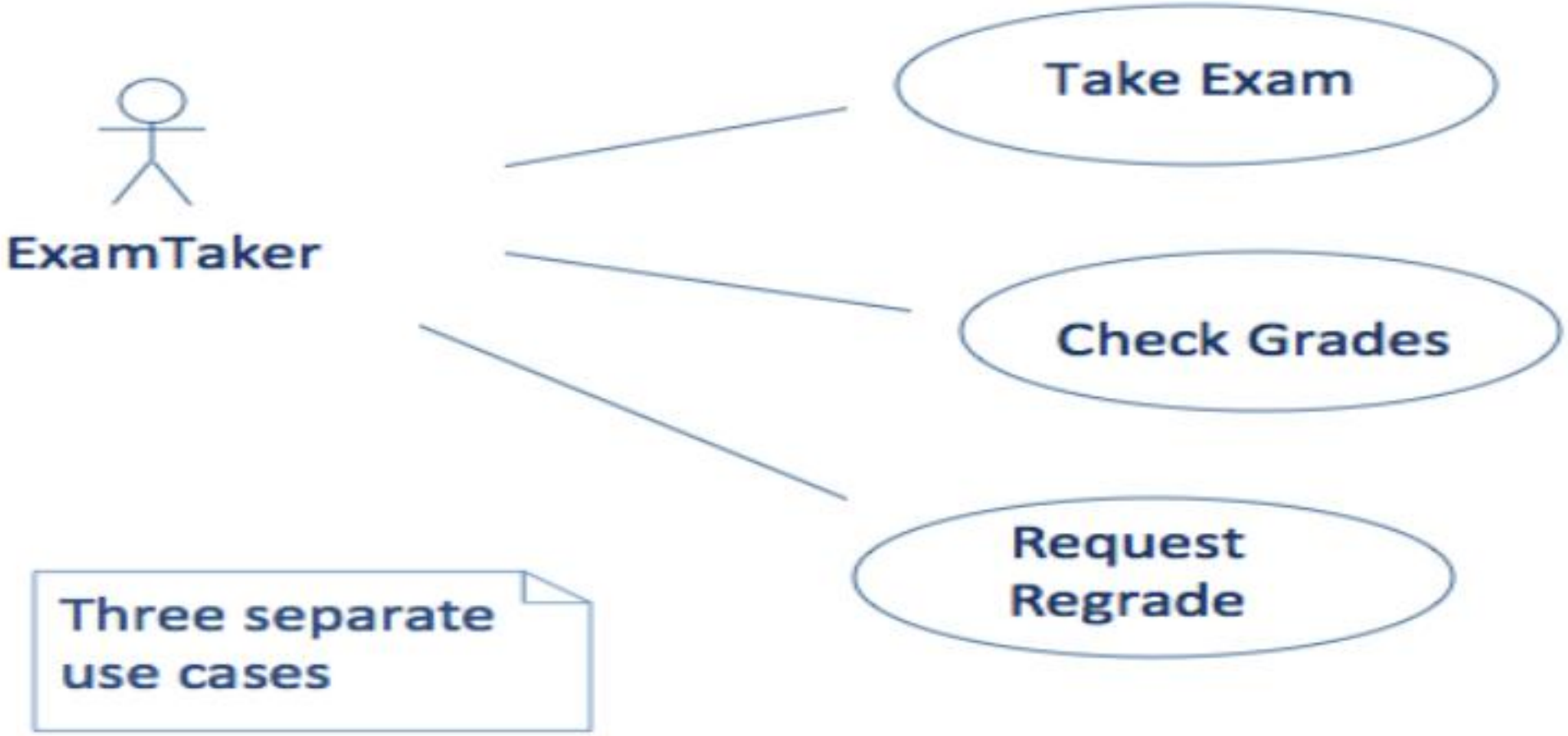
- Actor is **role**, not an individual
  (e.g., librarian can have many roles)
- Actor must be a **beneficiary** of the use case
  (e.g., not librarian who processes book when borrowed)

In naming actors, choose names that describe the role, not generic names, such as "user" or "client".

Example: Two Simple Use Cases



Use Cases for Exam System



Describing a Use Case

Some organizations have complex documentation standards for describing a use case.

At the very least, the description should include:

- The name of the use case, which should summarize its purpose
- The actor or actors
- The flow of events
- Assumptions about entry conditions

Outline of Take Exam Use Case

Name of Use Case: Take Exam

Actor(s): ExamTaker

Flow of events:

1. ExamTaker connects to the Exam server.
2. Exam server checks whether ExamTaker is already authenticated and runs authentication process if necessary.
3. ExamTaker selects a exam from a list of options.
4. ExamTaker repeatedly selects a question and either types in a solution, attaches a file with a solution, edits a solution or attaches a replacement file.

Outline of Take Exam Use Case Cont.

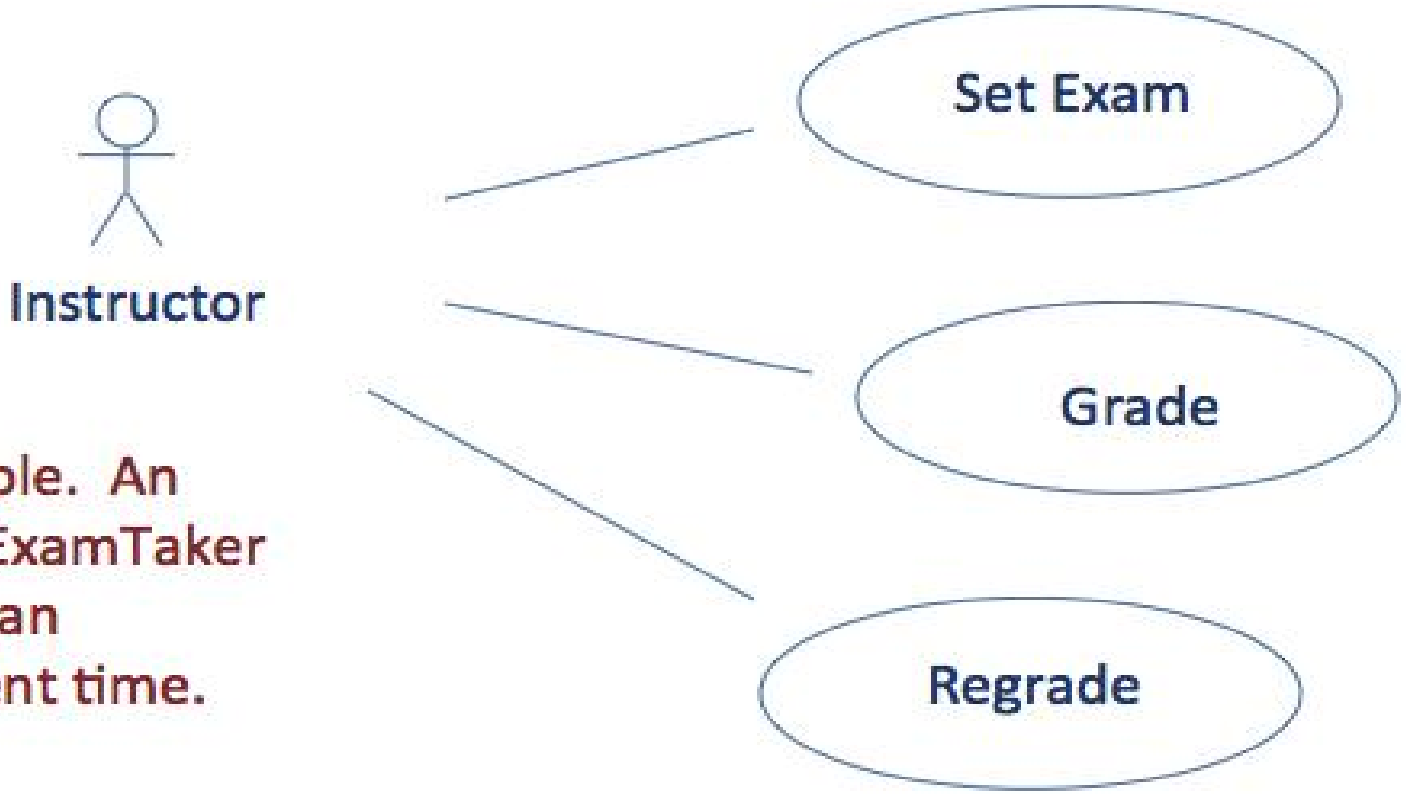
Flow of events (continued):

5. ExamTaker either submits completed exam or saves current state.
6. When a completed exam is submitted, Exam server checks that all questions have been attempted and either sends acknowledgement to ExamTaker, or saves current state and notifies ExamTaker of incomplete submission.
7. ExamTaker logs out.

Entry conditions:

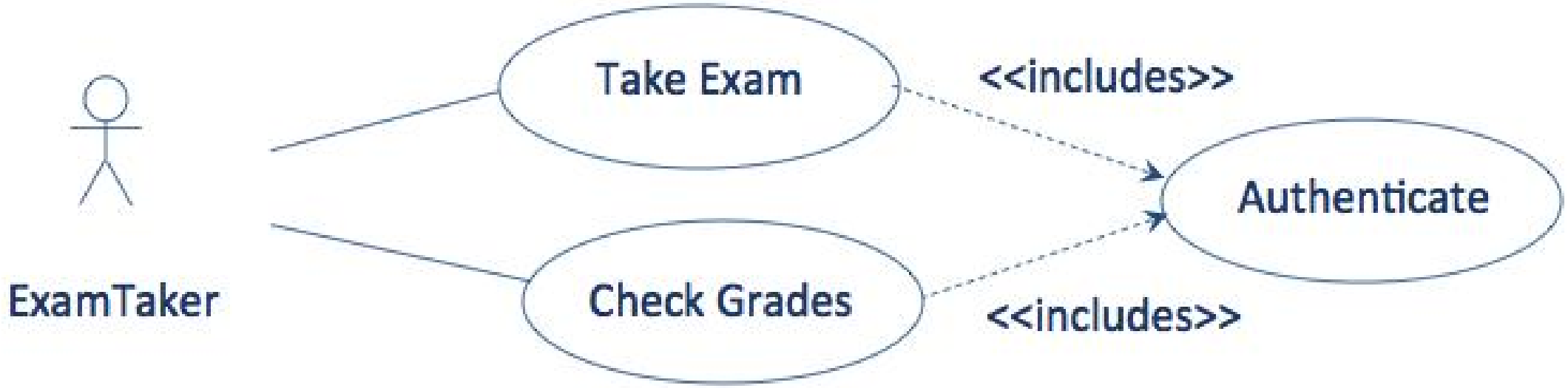
1. ExamTaker must have authentication credentials.
2. Computing requirements: supported browser.

Take Exam Use Case Cont.



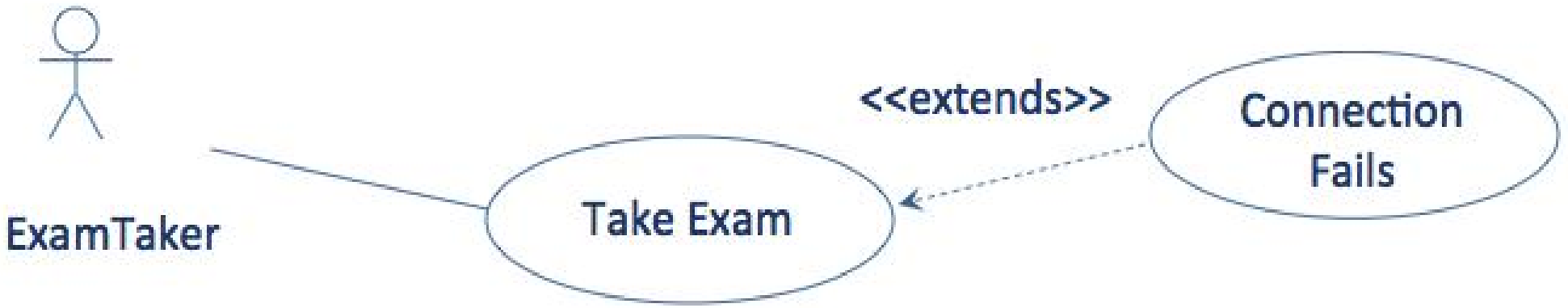
Note that actor is a role. An individual can be an ExamTaker on one occasion and an Instructor at a different time.

Relationship Between Use Cases: <<includes>>



The Authenticate use case may be used in other contexts

Relationship Between Use Cases: <<extends>>



<<includes>>	is used for use cases that are in the flow of events of the main use case.
<<extends>>	is used for exceptional conditions, especially those that can occur at any time.

Use cases for the MHC-PMS

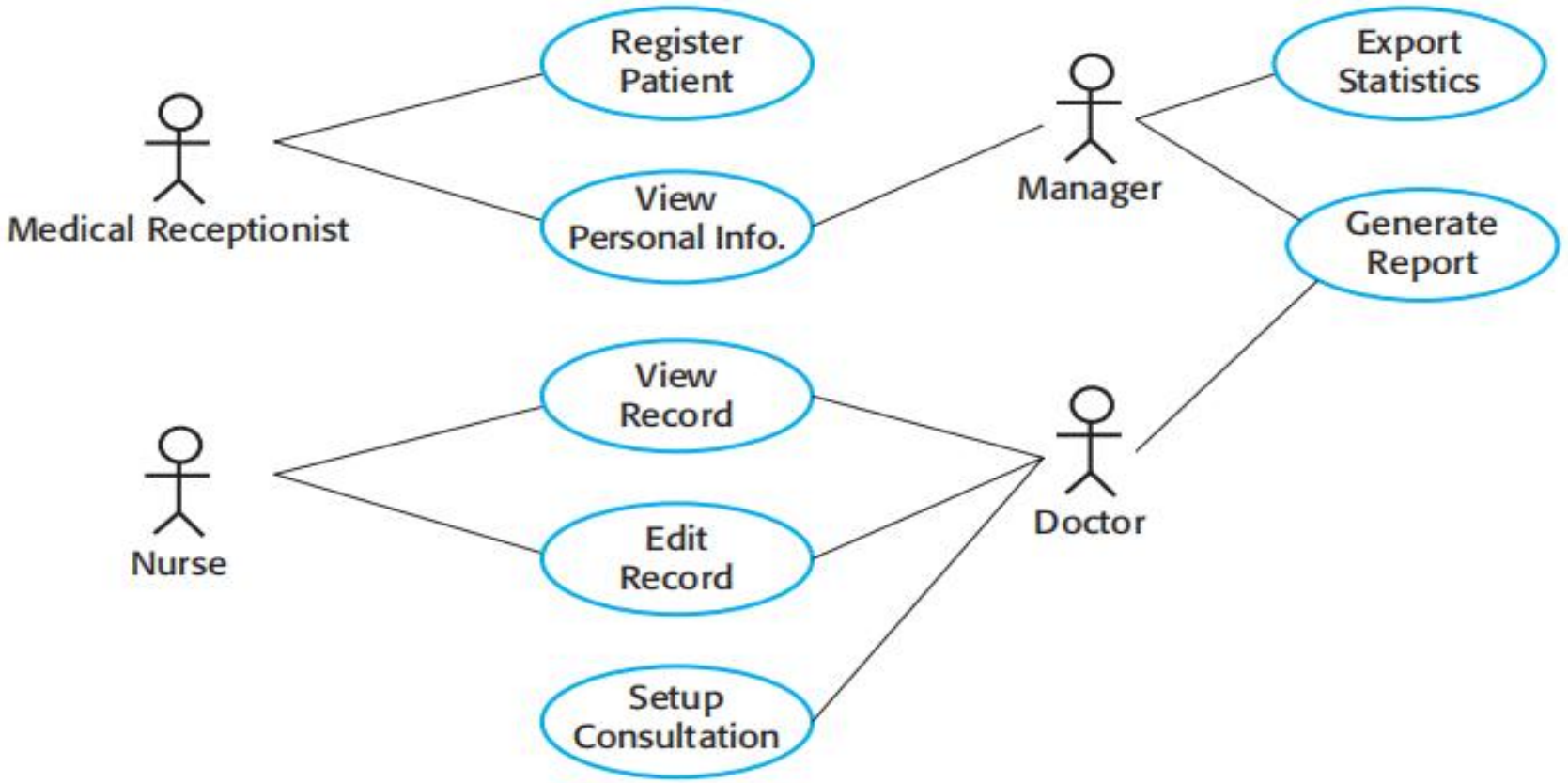


Figure 4.15 Use cases for the MHC-PMS

Scenarios and Use Cases in Development Cycle

Scenarios and use cases are both intuitive -- easy to discuss with clients

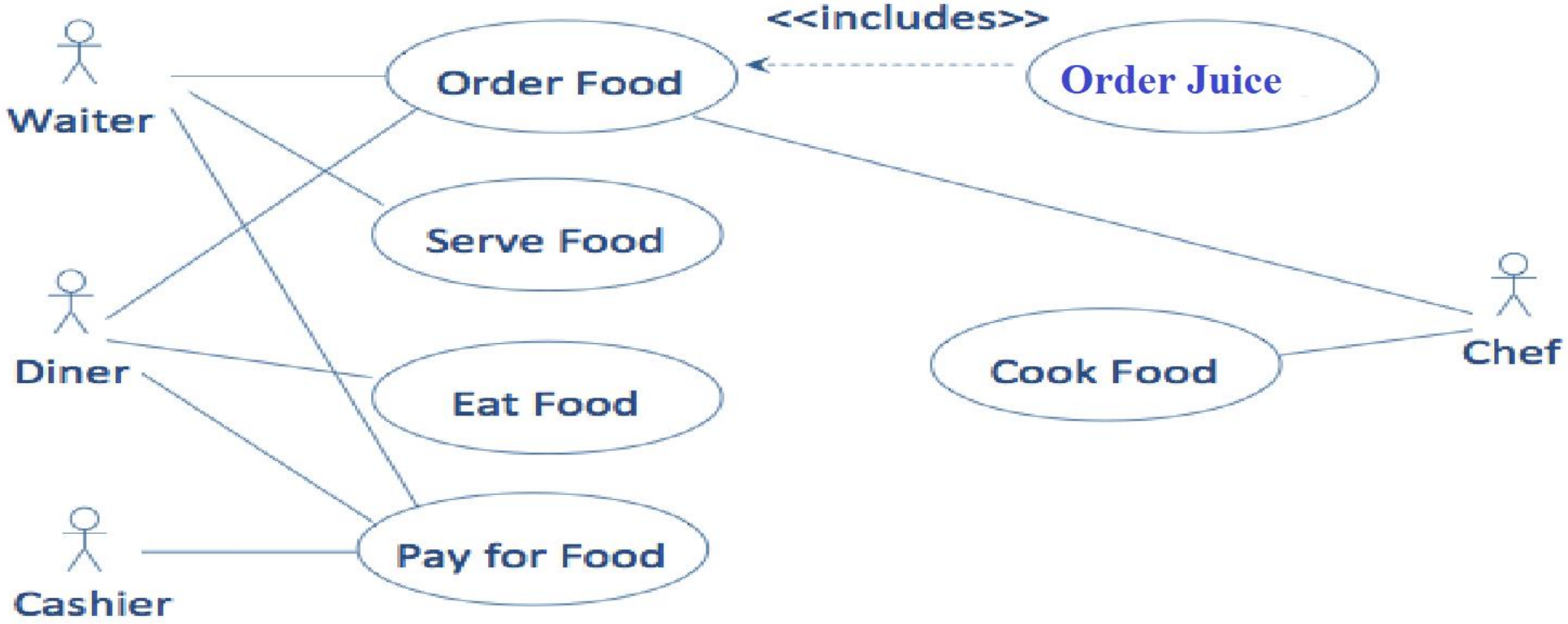
Scenarios are a tool for requirements analysis.

- They are useful to validate use cases and in checking the design of a system.
- They can be used as test cases for acceptance testing.

Use cases are a tool for modeling requirements.

- A set of use cases can provide a framework for the requirements specification.
- Use cases are the basis for system and program design, but are often hard to translate into class models

Use Case with Several Actors



This restaurant example is based on a use case diagram from Wikipedia.

Use case approach

- **A use case describes a sequence of interactions between a system and an external actor that results in some outcome that provides value to the actor.**
- An *actor* is a person (or sometimes another software system or a hardware device) that interacts with the system to perform a use case.
  - **For example**, the Chemical Tracking System's "Request a Chemical" use case involves an actor named *Requester*.
  - There is no CTS user class named Requester.
  - Both chemists and members of the chemical stockroom staff may request chemicals, so members of either user class may perform the Requester role.

Tabular Format and guideline to write use case(s)

Use Case ID:	Enter a unique numeric identifier for the Use Case. e.g. UC-1.2.1
Use Case Name:	Enter a short name for the Use Case using an active verb phrase. e.g. Withdraw Cash
Actors:	[An actor is a person or other entity external to the software system being specified who interacts with the system and performs use cases to accomplish tasks. **Different actors often correspond to different user classes, or roles, identified from the customer community that will use the product**. Name the actor that will be initiating this use case (primary) and any other actors who will participate in completing the use case (secondary).]
Description:	[Provide a brief description of the reason for and outcome of this use case.]
Level:	Enter Use Case Goal Level here: High/Medium/Low
Trigger:	[Identify the event that initiates the use case. This could be an external business event or system event that causes the use case to begin, or it could be the first step in the normal flow.]

Tabular Format and guideline to write use case(s)

Preconditions:	[List any activities that must take place, or any conditions that must be true, before the use case can be started. Number each pre-condition. e.g. 1. Customer has active deposit account with ATM privileges 2. Customer has an activated ATM card.]
Includes:	[List any other use cases that are included ("called") by this use case. Common functionality that appears in multiple use cases can be split out into a separate use case that is included by the ones that need that common functionality. e.g. steps 1-4 in the normal flow would be required for all types of ATM transactions- a Use Case could be written for these steps and "included" in all ATM Use Cases.]

Tabular Format and guideline to write use case(s)

Normal Flow:	[Provide a detailed description of the user actions and system responses that will take place during execution of the use case under normal, expected conditions. This dialog sequence will ultimately lead to accomplishing the goal stated in the use case name and description. 1. Customer inserts ATM card 2. Customer enters PIN 3. System prompts customer to enter language performance English or Spanish 4. System validates if customer is in the bank network 5. System prompts user to select transaction type 6. Customer selects Withdrawal From Checking 7. System prompts user to enter withdrawal amount 8. … 9. System ejects ATM card]

Tabular Format and guideline to write use case(s)

Alternative Flows: [Alternative Flow 1 – Not in Network]	[Document legitimate branches from the main flow to handle special conditions (also known as extensions). For each alternative flow reference the branching step number of the normal flow and the condition which must be true in order for this extension to be executed. e.g. Alternative flows in the Withdraw Cash transaction: 4a. In step 4 of the normal flow, if the customer is not in the bank network 1. System will prompt customer to accept network fee 2. Customer accepts 3. Use Case resumes on step 5 4b. In step 4 of the normal flow, if the customer is not in the bank network 1. System will prompt customer to accept network fee 2. Customer declines 3. Transaction is terminated 4. Use Case resumes on step 9 of normal flow Note: Insert a new row for each distinctive alternative flow.]

Tabular Format and guideline to write use case(s)

Exceptions:	[Describe any anticipated error conditions that could occur during execution of the use case, and define how the system is to respond to those conditions. e.g. Exceptions to the Withdraw Case transaction 2a. In step 2 of the normal flow, if the customer enters and invalid PIN 1. Transaction is disapproved 2. Message to customer to re-enter PIN 3. Customer enters correct PIN 4. Use Case resumes on step 3 of normal flow]
Post conditions:	[Describe the state of the system at the end of the use case execution. Should include both *minimal guarantees* (what must happen even if the actor's goal is not achieved) and the *success guarantees* (what happens when the actor's goal is achieved. Number each post-condition. e.g. 1. Customer receives cash Customer account balance is reduced by the amount of the withdrawal and transaction fees]

Tabular Format and guideline to write use case(s)

Special/Quality Requirements (other information):	[Identify any additional requirements, such as nonfunctional requirements, for the use case that may need to be addressed during design or implementation. These may include performance requirements or other quality attributes.]
Assumptions:	[List any assumptions that were made in the analysis that led to accepting this use case into the product description and writing the use case description. e.g. For the Withdraw Cash Use Case, an assumption could be: The Bank Customer understands either English or Spanish language.]
Business Rule:	Some business rules constrain which roles can perform all or parts of a use case. Perhaps only users who have certain privilege levels can perform specific alternative flows. That is, the rule might impose preconditions that the system must test before letting the user proceed. Business rules can influence specific steps in the normal flow by defining valid input values or dictating how computations are to be performed.

Tabular Use Case Example-1

Use Case ID:	UC-1.1
Use Case Name:	Displaying Side Menu
Actors:	Dumb/deaf
Description:	In this use case, side menu will be displayed.
Trigger:	Clicking on side menu button will initiate the process.
Preconditions:	Side menu button should be pressed.
Post conditions:	Side menu will be displayed.
Normal Flow:	1. Opening the application. 2. Clicking on the side menu button. 3. Side menu will display.
Alternative Flows:	1. Opening the application. 2. Swipe left to right on screen 3. Side menu will display.
Exceptions:	In step 2 of normal flow if user click on right side of screen 1. Side menu will disappear. 2. Main interface will appear. In step 2 of normal flow if user swipe right to left 1. Side menu will disappear. 2. Main interface will appear.
Includes:	NA
Assumptions:	Application will be installed.
Notes and Issues:	NA

Tabular Use Case Example-2

ID and Name:	UC-4 Request a Chemical		
Created By:	Lori	Date Created:	8/22/13
Primary Actor:	Requester	Secondary Actors:	Buyer, Chemical Stockroom, Training Database
Description:	The Requester specifies the desired chemical to request by entering its name or chemical ID number or by importing its structure from a chemical drawing tool. The system either offers the Requester a container of the chemical from the chemical stockroom or lets the Requester order one from a vendor.		
Trigger:	Requester indicates that he wants to request a chemical.		
Preconditions:	PRE-1. User's identity has been authenticated. PRE-2. User is authorized to request chemicals. PRE-3. Chemical inventory database is online.		
Postconditions:	POST-1. Request is stored in the CTS. POST-2. Request was sent to the Chemical Stockroom or to a Buyer.		
Normal Flow:	**4.0 Request a Chemical from the Chemical Stockroom** 1. Requester specifies the desired chemical. 2. System lists containers of the desired chemical that are in the chemical stockroom, if any. 3. System gives Requester the option to View Container History for any container. 4. Requester selects a specific container or asks to place a vendor order (see 4.1). 5. Requester enters other information to complete the request. 6. System stores the request and notifies the Chemical Stockroom.		
Alternative Flows:	**4.1 Request a Chemical from a Vendor** 1. Requester searches vendor catalogs for the chemical (see 4.1.E1). 2. System displays a list of vendors for the chemical with available container sizes, grades, and prices. 3. Requester selects a vendor, container size, grade, and number of containers. 4. Requester enters other information to complete the request. 5. System stores the request and notifies the Buyer.		
Exceptions:	**4.1.E1 Chemical Is Not Commercially Available** 1. System displays message: No vendors for that chemical. 2. System asks Requester if he wants to request another chemical (3a) or to exit (4a). 3a. Requester asks to request another chemical. 3b. System starts normal flow over. 4a. Requester asks to exit. 4b. System terminates use case.		
Priority:	High		
Frequency of Use:	Approximately 5 times per week by each chemist, 200 times per week by chemical stockroom staff		
Business Rules:	BR-28, BR-31		
Other Information:	The system must be able to import a chemical structure in the standard encoded form from any of the supported chemical drawing packages.		
Assumptions:	Imported chemical structures are assumed to be valid.		

Template to write Functional Requirement

Identifier	Requirement ID
Title	Title of requirement
Requirement	Description of requirement
Source	Where requirement come from (who elicitate it)
Rationale	Motivation behind the requirement
Restrictions and Risk	Any restriction and risk that requirement must be fulfilled
Dependencies	Requirements ID that are dependent on this requirement
Priority	High/Medium/Low

Functional Requirement – Example-1

Identifier	1.2.1
Title	Edit Application
Requirement	The students must be able to access the main application tab so that they can edit their application and complete all the requirements of the application in order to apply to the universities they intends to
Source	Team Member
Rationale	To ensure that the student provides all the necessary details required
Restrictions and Risk	All the fields of the application have to be full filled in order to submit an application
Dependencies	1.1.1
Priority	High

Functional Requirement – Example-2

Table 8: Validate Email

Identifier	FR1.1.8
Title	Validate Email
Requirement	System will be able to validate email address entered by the user.
Source	Supervisor
Rationale	To inform the user if he/she enters an invalid email address
Business Rule (if required)	User must enter valid email address.
Dependencies	FR1.1.2
Priority	High

Functional Requirement – Example-3

Table 2: Perform Animation on Model

Identifier	FR-2.1.4
Title	Perform Animation on Model
Requirement	Baby model will perform the selected animation, which will be used to train the parents.
Source	Supervisor
Rationale	Baby's model must perform the selected animation
Business Rule (if required)	N/A
Dependencies	N/A
Priority	High

Functional Requirement – Example-4

Identifier	1.1.1
Title	Login/Register
Requirement	A student should be able to register and log in to the system
Source	Teacher
Rationale	Without login or registration the student cannot use the system
Restrictions and Risk	The student must remember the credentials and keep it safe
Dependencies	1.1.2
Priority	High

Requirements Validation

Requirements validation

- **Requirements validation** examines the specification to ensure that all software requirements have been stated unambiguously; that inconsistencies, omissions, and errors have been detected and corrected; and that the work products conform to the standards established for the process, the project, and the product.
- **Concerned with demonstrating that the requirements** define the system that the customer really wants.
- **Requirements error costs are high so validation is very important**
  - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error.

Requirements checking

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

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

- **A requirements review is a process where a group of people from the system customer and the system developer read the requirements document in detail and check for errors, anomalies, and inconsistencies. Once these have been detected and recorded, it is then up to the customer and the developer to negotiate how the identified problems should be solved.**
- **Regular reviews** should be held while the requirements definition is being formulated.
- **Both client and contractor staff should be involved in revie**ws.
- **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

- **Requirements management is the process of managing changing** requirements during the requirements engineering process and system development.
- **New requirements emerge as a system** is being developed and after it has gone into use.
- **You need to keep track of individual requirements and maintain links between dependent requirements so that you can assess the impact of requirements changes**. You need to establish a formal process for making change proposals and linking these to system requirements.

Requirements Change

- **The priority of requirements from different viewpoints changes** during the development process.
- **System customers may specify requirements** from a business perspective that conflict with end-user requirements.
- **The business and technical environment** of the system changes during its development.

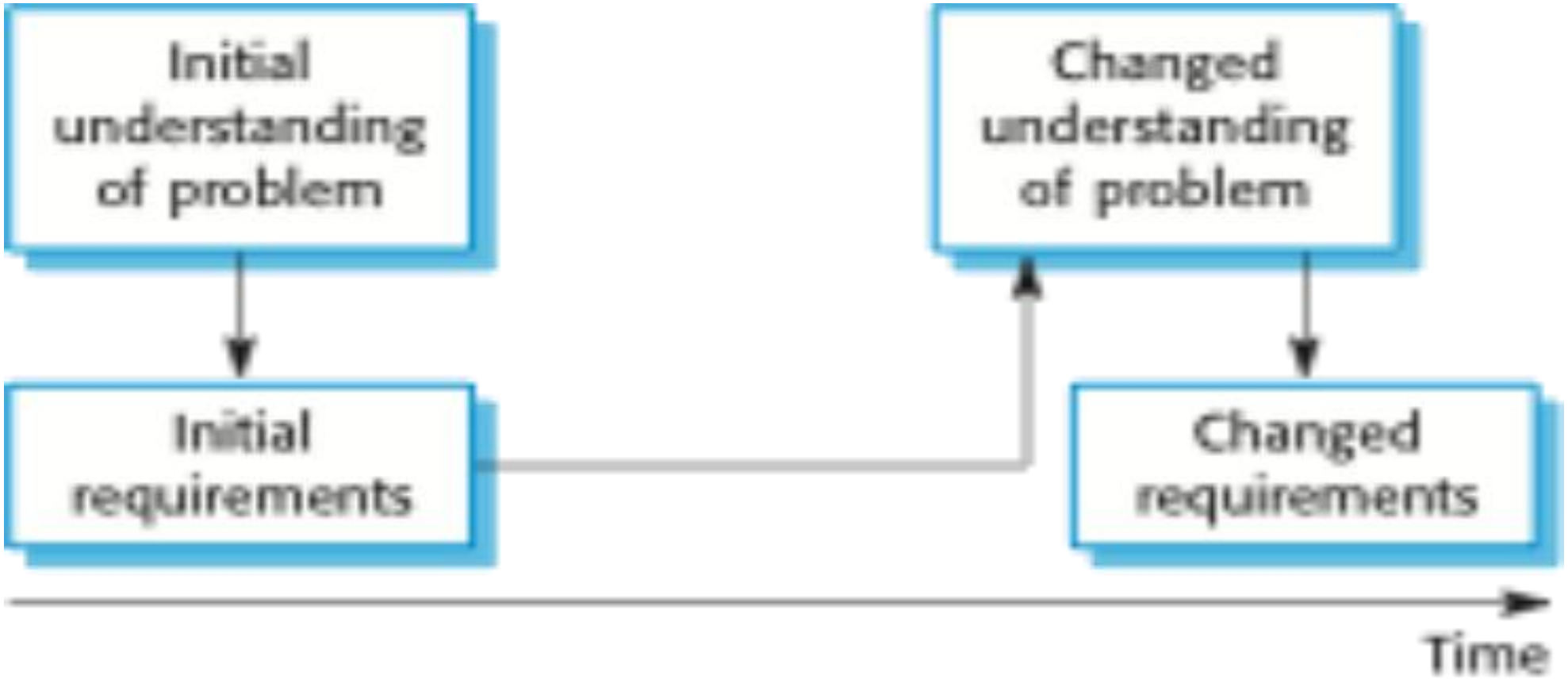
Changing Requirements

- **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.

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 requester 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.

Requirements Management Planning

- During the requirements engineering process, you have to plan:
  - Requirements identification
    - How requirements are individually identified;
  - A change management process
    - The process followed when analyzing a requirements change;
  - Traceability policies
    - The amount of information about requirements relationships that is maintained;
  - CASE tool support
    - The tool support required to help manage requirements change;

Traceability

- **Requirements traceability** You need to keep track of the relationships between requirements, their sources, and the system design so that you can analyze the reasons for proposed changes and the impact that these changes are likely to have on other parts of the system.
- Traceability is concerned with the relationships between requirements, their sources and the system design
- **Source traceability**
  - Links from requirements to stakeholders who proposed these requirements;
- **Requirements traceability**
  - Links between dependent requirements;
- **Design traceability**
  - Links from the requirements to the design;

A traceability matrix

Req. id	1.1	1.2	1.3	2.1	2.2	2.3	3.1	3.2
1.1		D	R					
1.2			D			D		D
1.3	R			R				
2.1			R		D			D
2.2								D
2.3		R		D				
3.1								R
3.2							R	

CASE Tool Support

- Requirements storage
  - Requirements should be managed in a secure, managed data store.
- Change management
  - The process of change management is a workflow process whose stages can be defined and information flow between these stages partially automated.
- Traceability management
  - Automated retrieval of the links between requirements.

Requirements Change management

- Should apply to all proposed changes to the requirements.
- Principal stages
  - Problem analysis. Discuss requirements problem and propose change;
  - Change analysis and costing. Assess effects of change on other requirements;
  - Change implementation. Modify requirements document and other documents to reflect change.

Requirements Change Management

Enduring and Volatile Requirements

- Enduring requirements. Stable requirements derived from the core activity of the customer organization. E.g. a hospital will always have doctors, nurses, etc. May be derived from domain models
- Volatile requirements. Requirements which change during development or when the system is in use. In a hospital, requirements derived from health-care policy

Summary

- **The requirements engineering process** includes a gathering, requirements elicitation and analysis, requirements specification and requirements management.
- **Requirements elicitation and analysis** is iterative involving domain understanding, requirements collection, classification, structuring, prioritization and validation.
- Systems have multiple stakeholders with different requirements.

Summary

- **Requirements validation** is concerned with checks for validity, consistency, completeness, realism and verifiability.
- **Business changes** inevitably lead to changing requirements.
- **Requirements management** includes planning and change management.

Class TASK

- Consider your watch as a system and set the time 2 minutes ahead. Write down each interaction between you and your watch as a scenario. Record all interactions, including any feedback the watch provides you

Home Tasks

- Draw a use case diagram for a ticket distributor for a train system.
- The system includes two actors: a traveler who purchases different types of tickets, and a central computer system that maintains a reference database for the tariff.
- **Use cases should include**
  - BuyOneWayTicket,
  - BuyWeeklyCard,
  - BuyMonthlyCard, and
  - UpdateTariff.
- **Also include the following exceptional cases:**
  - TimeOut (i.e., traveler took too long to insert the right amount),
  - TransactionAborted (i.e., traveler selected the cancel button without completing the transaction),
  - DistributorOutOfChange, and
  - DistributorOutOfPaper


Thank
you!