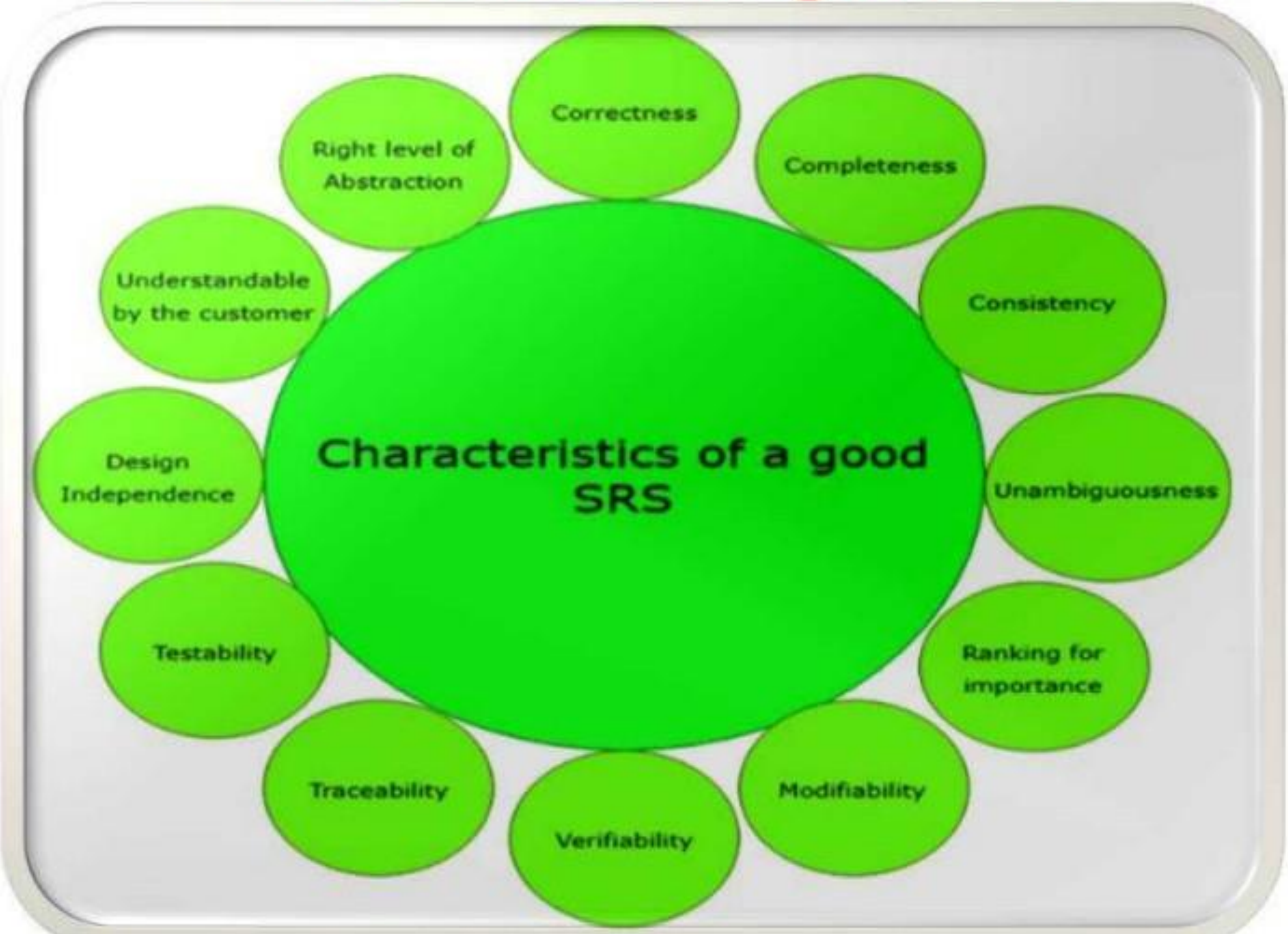
Chapter 3

Software Requirement Analysis & Specifications

Software Requirement Specification (SRS)

- SRS is the medium through which the client and the user needs are accurately specified.
- **Software requirement** is the description and specifications of a system.
- SRS is a document that completely describes **WHAT** the proposed system should do without describing **HOW** the software will do it.
- The basic goal of requirement phase is to produce the SRS, which describes the complete external behavior of the proposed software.
- A high quality SRS is a pre-requisite to high quality software.
- A high quality SRS reduces the development cost.

Characteristics of good SRS



What should the SRS address?

The IEEE standard, the basic issues that the SRS writer(s) shall address are the following:

- **Functionality:** What is the software supposed to do?
- **External interfaces:** How does the software interact with people, the system's hardware, other hardware, and other software?
- **Performance.:** What is the speed, availability, response time, recovery time of various software functions, etc.?
- **Attributes:** What are the portability, correctness, maintainability, security, etc. considerations?
- **Design constraints imposed on an implementation:** Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s) etc.?

Definition of Requirement

- ***"Requirement is the descriptions and specifications of a system".***
- Software requirements analysis is necessary to avoid creating a software product that fails to meet the customer's needs.
- Data, functional, and behavioral requirements are elicited from the customer and refined to create a specification that can be used to design the system.
- Software requirements work products must be reviewed for clarity, completeness, and consistency.

Types of Requirements

- **User requirements**

  Statements in natural language plus diagrams of the services the system provides and its operational constraints. ***Written for customers.***

- **System requirements**

  A structured document setting out detailed descriptions of the system services. ***Written as a contract between client and contractor.***

- **Software specification**

  A detailed software description which can serve as a basis for a design or implementation. ***Written for developers.***

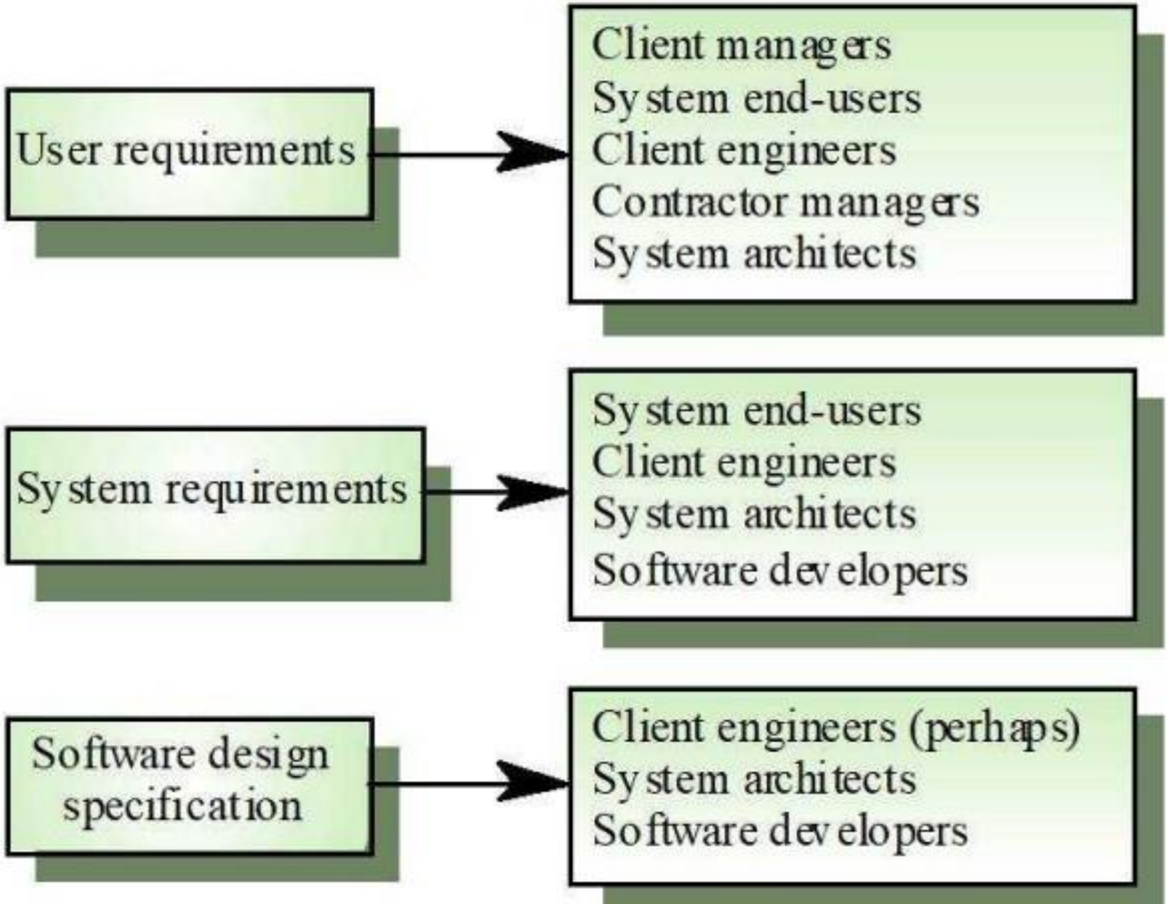
- **Domain requirements**

  Requirements that come from the application domain of the system and that reflect characteristics of that domain

  May be new functional requirements, constraints on existing requirements or define specific computations

  If domain requirements are not satisfied, the system may be unworkable

Requirements readers



Functional and non-functional requirements

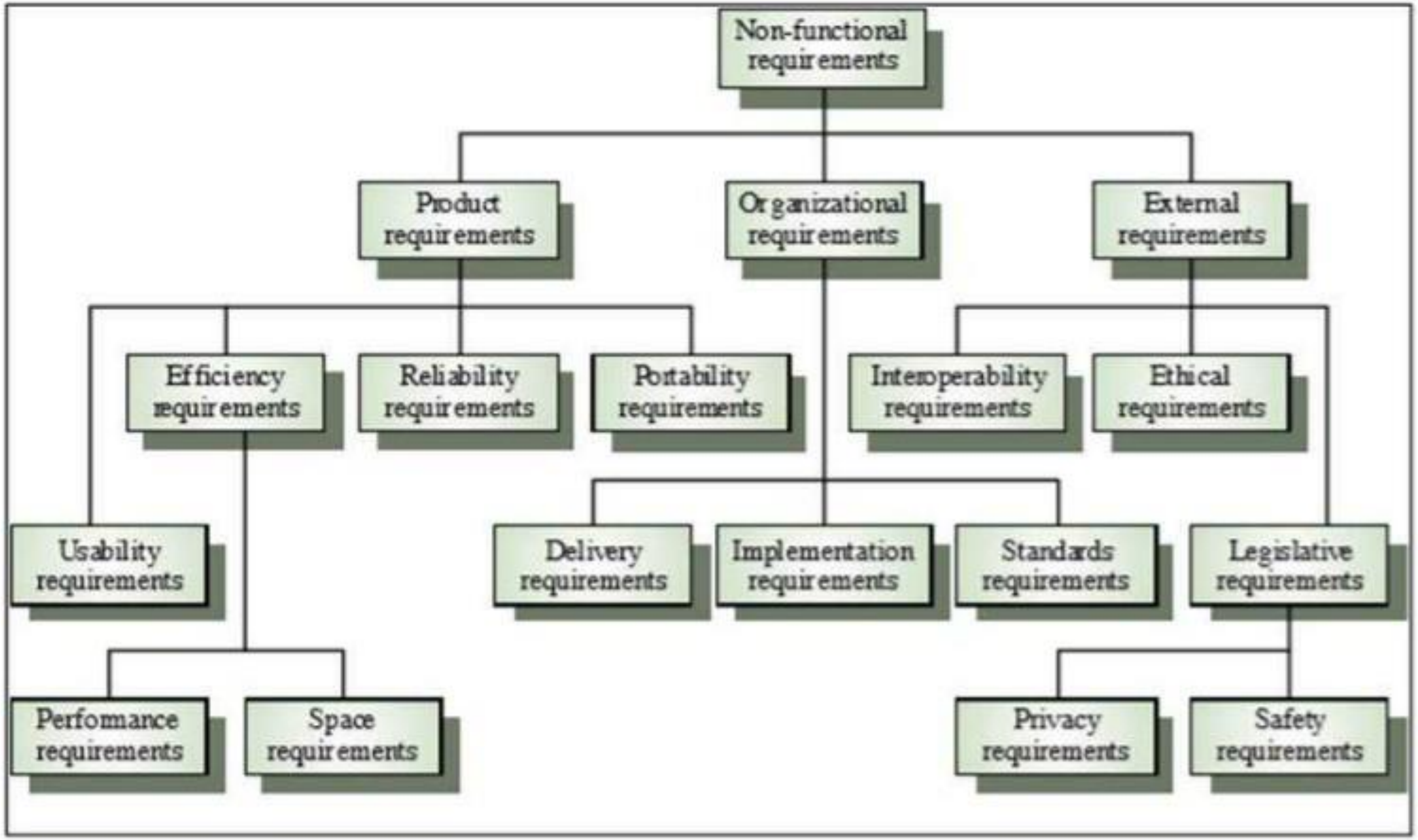
- **Functional requirements**
- Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
- Describe functionality or system services
- Depend on the type of software, expected users and the type of system where the software is used
- Functional user requirements may be high-level statements of what the system should do but functional system requirements should describe the system services in detail
- **Non-functional requirements**
- constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
- Define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless

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 requirements define the overall qualities or attributes of the resulting system"

Non-functional requirement types



Requirements describe

What not **How**

Produces one large document written in natural language contains a description of what the system will do without describing how it will do it.

Requirements are difficult to uncover

- Requirements change
- Problem of scope
- Problem of understanding
- Tight project Schedule
- Communication barriers
- Market driven software development
- Lack of resources

Requirement Engineering

- Requirements Engineering provides the appropriate mechanism for **understanding what the customer wants**, **analyzing the need**, **assessing feasibility**, **negotiating a reasonable solution**, **specifying the solution unambiguously**, **validating the specification**, and **managing the requirements** as they are transformed into a operational system.
- Requirement Engineering is the **disciplined application** of proven principles, methods, tools, and notations **to describe a proposed system's intended behavior and its associated constraints**.
- **SRS may act as a contract between developer and customer.**

Requirement Engineering Process

It is the processes used to discover, analyze and validate system requirements. 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 can be described in **six** distinct steps:

Step 1: **Requirements elicitation**

Step 2: **Requirements analysis and negotiation**

Step 3: **Requirements specification**

Step 4: **System modeling**

Step 5: **Requirements validation**

Step 6: **Requirements management**

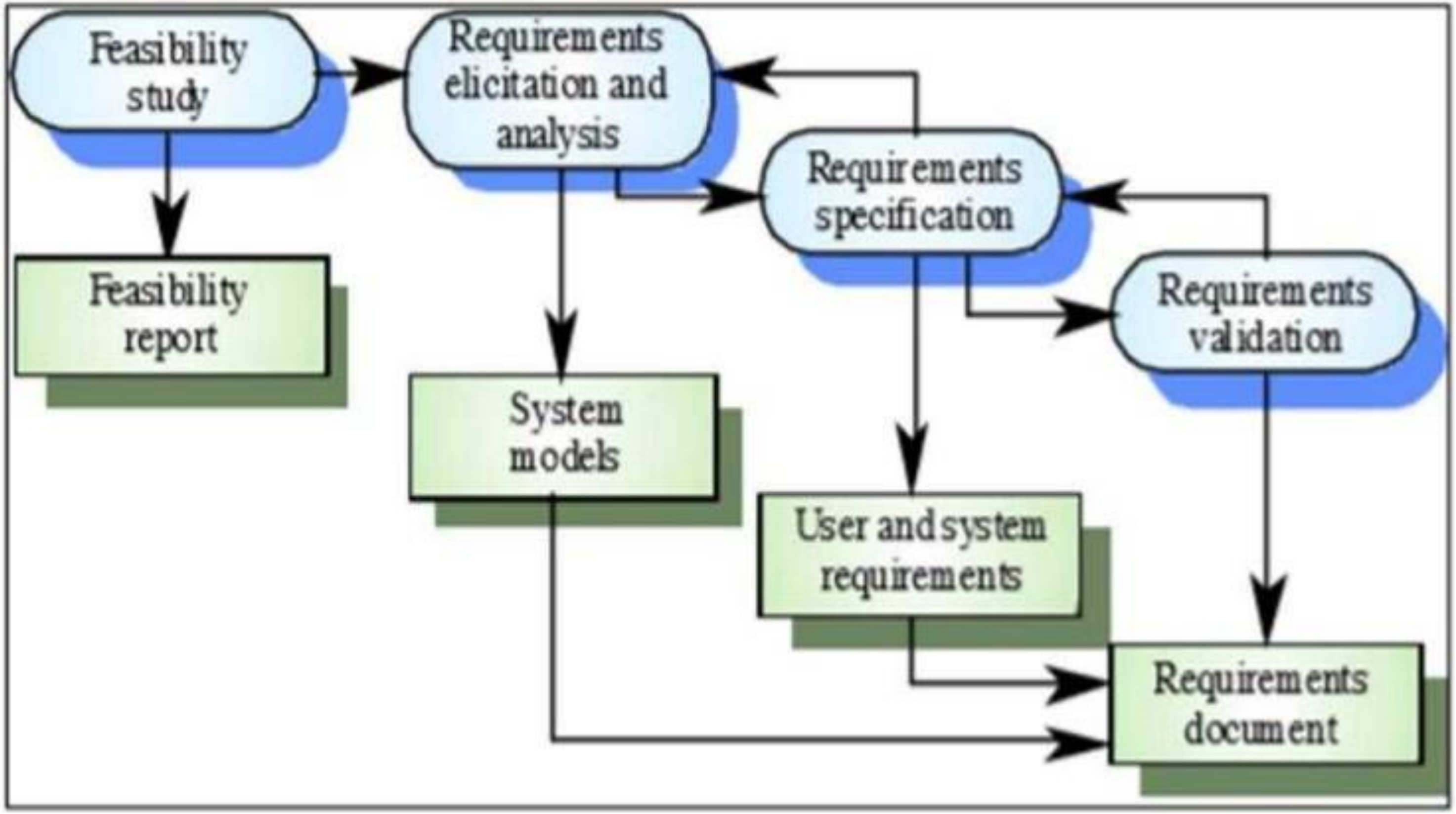
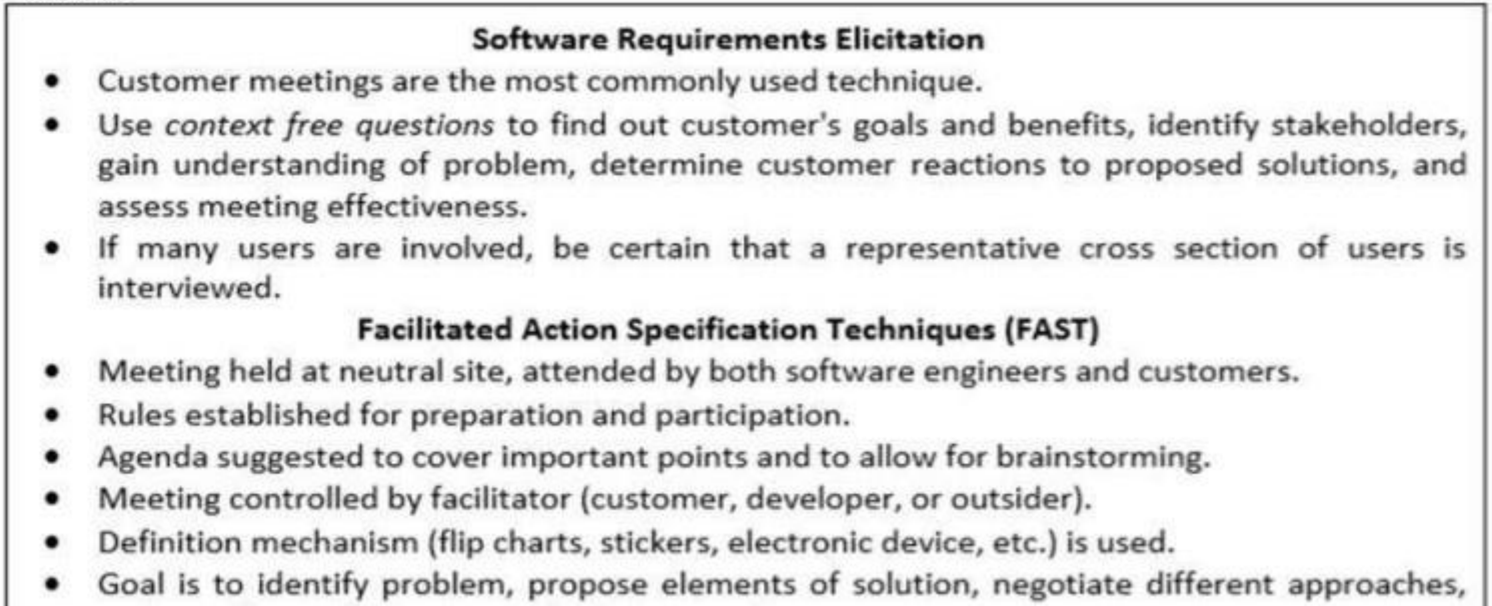


Fig: Requirement Engineering Process

Requirement Engineering Process

Step 1: Requirements elicitation

- It is the practice of obtaining the requirements of a system from users, customers and other stakeholders. The practice is also sometimes referred to as **Requirement gathering**.
- Find out from customers what the product objectives are, what is to be done, how the product fits into business needs, and how the product is used on a day to day basis.



Software Requirements Elicitation

- Customer meetings are the most commonly used technique.
- Use *context free questions* to find out customer's goals and benefits, identify stakeholders, gain understanding of problem, determine customer reactions to proposed solutions, and assess meeting effectiveness.
- If many users are involved, be certain that a representative cross section of users is interviewed.

Facilitated Action Specification Techniques (FAST)

- Meeting held at neutral site, attended by both software engineers and customers.
- Rules established for preparation and participation.
- Agenda suggested to cover important points and to allow for brainstorming.
- Meeting controlled by facilitator (customer, developer, or outsider).
- Definition mechanism (flip charts, stickers, electronic device, etc.) is used.
- Goal is to identify problem, propose elements of solution, negotiate different approaches,

Requirements elicitation practice include the following:

- Interviews
- Questionnaires
- User observation
- Workshops
- Brain storming
- Use cases
- Role playing
- And prototyping

Problems of Requirement Elicitation

- **Problems of scope:** The boundary of system is ill-defined. Or unnecessary details are provided.
- **Problems of understanding:** The users are not sure of what they need, and don't have full understanding of the problem domain.
- **Problems of volatility:** the requirements change overtime.

Elicitation problems Ctd...

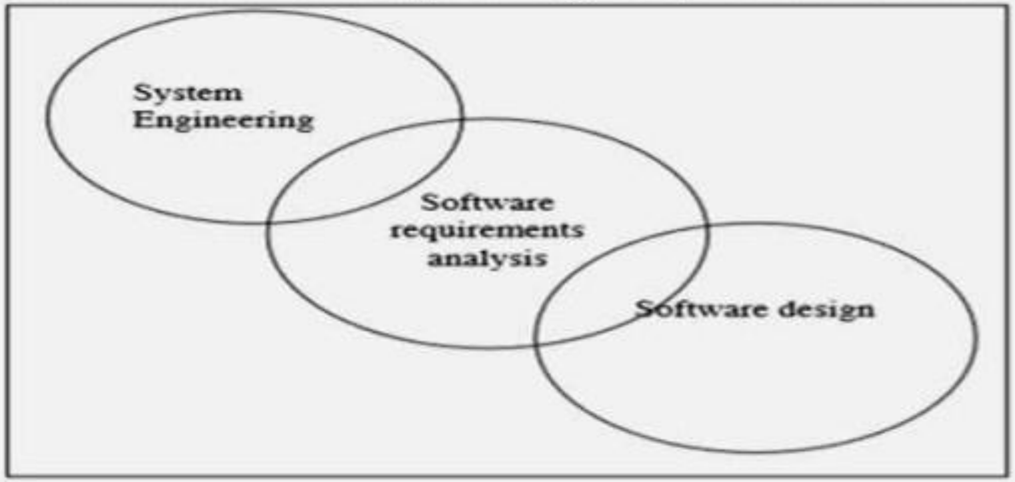
- Not having good skills to collect information and cannot understand the group problem.
- Not well defined the actual Problem.
- Not focus on the requirement of the system but more on the design which is useless on this stage.
- The requirement should be changeable according to time.
- Lack of analyst knowledge with the problem.
- Lack of user's knowledge also creates problems for elicitation.
- Problem with understanding the language between user and analyst.
- Ignoring or omitting the actual problem.
- The boundary of the problem should be well defined

Guidelines of Requirements Elicitation

- Assess the business and technical feasibility for the proposed system
- Identify the people who will help specify requirements.
- Define the technical environment (e.g. computing architecture, operating system, telecommunication needs) into which the system or product will be placed
- Identify “domain constraints” (i.e. characteristics of the business environment specific to the application domain) that limit the functionality or performance of the system or product to build
- Define one or more requirements elicitation methods (e.g. interviews, team meetings, ..etc)
- Solicit participation from many people so that requirements are defined from different point of views.
- Create usage scenarios of use cases to help customers/ users better identify key requirements.

Step 2: Requirements analysis and negotiation

- Requirements are categorized and organized into subsets, relations among requirements identified, requirements reviewed for correctness, requirements prioritized based on customer needs.
- Software engineering task that bridges the gap between system level requirements engineering and software design.
- Provides software designer with a representation of system information, function, and behavior that can be translated to data, architectural, and component-level designs.
- Expect to do a little bit of design during analysis and a little bit of analysis during design.



Problems of requirements analysis

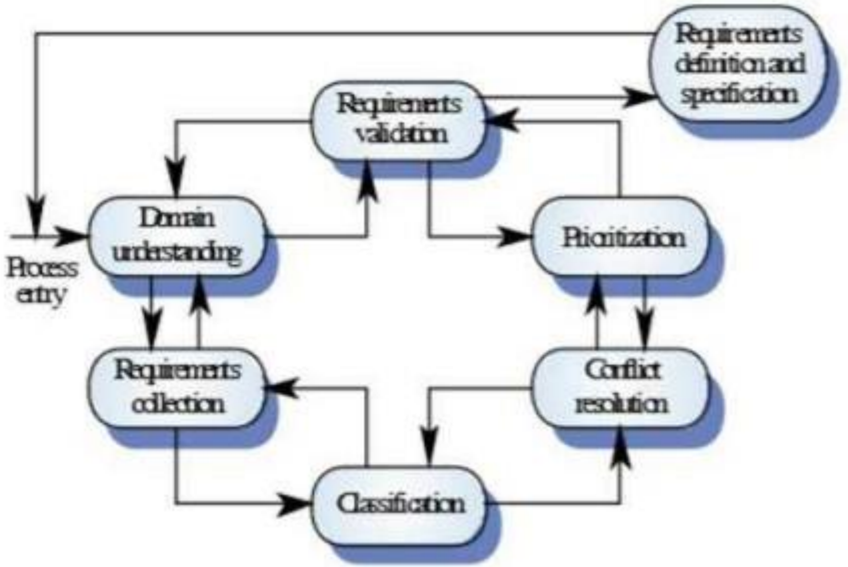
- Stakeholders don't know what they really want
- Stakeholders express requirements in their own terms
- Different stakeholders may have conflicting requirements
- Organizational and political factors may influence the system requirements
- The requirements change during the analysis process. New stakeholders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

Analysis Principles

- The information domain of the problem must be represented and understood. The functions that the software is to perform must be defined.
- Software behavior must be represented.
- Models depicting information, function, and behavior must be partitioned in a hierarchical manner that uncovers detail.
- The analysis process should move from the essential information toward implementation details

Requirements elicitation and analysis process

- 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.
- The process activities include:
  - ✓ Domain understanding
  - ✓ Requirements collection
  - ✓ Classification
  - ✓ Conflict resolution
  - ✓ Prioritization
  - ✓ Requirements checking



Step 3: Requirements specification

Specification Principles

- Separate functionality from implementation.
- Develop a behavioral model that describes functional responses to all system stimuli.
- Define the environment in which the system operates and indicate how the collection of agents will interact with it.
- Create behavioral model rather than an implementation model.
- Recognize that the specification must be extensible and tolerant of incompleteness.
- Establish the content and structure of a specification so that it can be changed easily.
- Representation format and content should be relevant to the problem.
- Representations should be revisable.

Software Requirement specification

- Requirements Specification is the direct result of a requirement analysis and can refer to:
  - Software Requirements Specification
  - Hardware Requirements Specification
- A Software Requirements Specification (SRS) – a requirements specification for a software system – is a complete description of the behavior of a system to be developed.
- It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements (such as performance requirements, quality standards, or design constraints)

Candidate format for software requirement specification:

✓ **Introduction**
✓ **Information Description**
✓ **Functional Description**
✓ **Behavioral Description**
✓ **Validation Criteria**
✓ **Bibliography and Appendix**

Step 4: **System modeling**

- system representation that shows relationships among the system components
- System modeling is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system.
- System modeling has now come to mean representing a system using some kind of graphical notation, which is now almost always based on notations in the Unified Modeling Language (UML).
- System molding helps the analyst to understand the functionality of the system and models are used to communicate with customers.
- Models are used during the requirements engineering process to help derive the requirements for a system, during the design process to describe the system to engineers implementing the system and after implementation to document the system's structure and operation. You may develop models of both the existing system and the system to be developed:

- Models of the existing system are used during requirements engineering. They help clarify what the existing system does and can be used as a basis for discussing its strengths and weaknesses. These then lead to requirements for the new system.
- Models of the new system are used during requirements engineering to help explain the proposed requirements to other system stakeholders. Engineers use these models to discuss design proposals and to document the system for implementation.

System Modeling Process

I. System Context Diagram
II. System Flow Diagram
III. System Specification (Developed by writing narrative descriptions for each subsystem)

Step 5: Requirements validation

- Requirements validation makes sure that requirements meet stakeholders' goals and don't conflict with them.
- examines the specification to ensure requirement quality and that work products conform to agreed upon standards.
- is the process of checking whether the requirements, as identified, do not contradict the expectations about the system of various stakeholders and do not contradict each other.
- It is Requirements Quality Control
- It checks for validity, consistency, completeness, realism, verifiability.

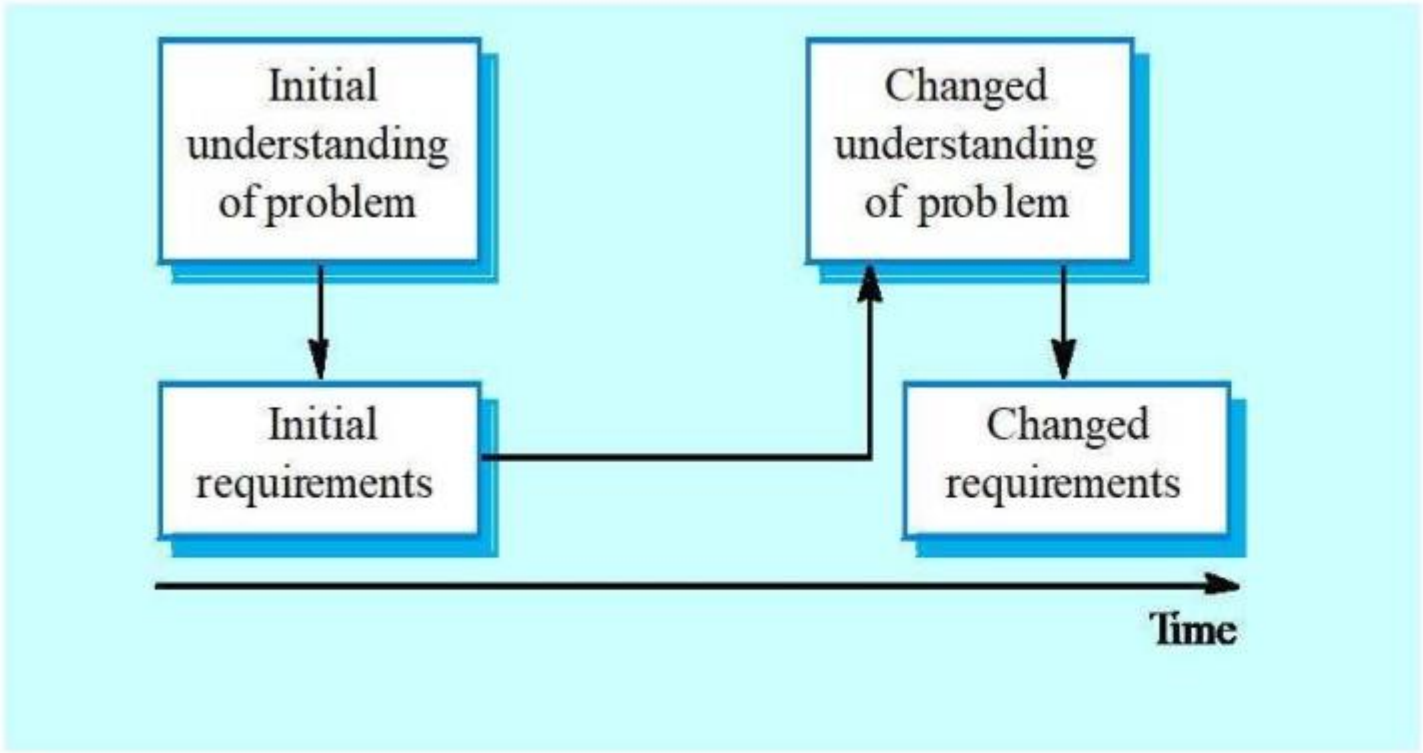
Requirements Validation techniques

1. **Requirements reviews**:- Systematic manual analysis of the requirements
2. **Prototyping** :- Using an executable model of the system to check requirements.
3. **Test-case generation** :- Developing tests for requirements to check testability
4. **Automated consistency analysis** :- Checking the consistency of a structured requirements description

Step 6: Requirements management

- Requirements management is the process of managing changing requirements during the requirements engineering process and system development.
- **Requirements are inevitably incomplete and inconsistent**
  - New requirements emerge during the process as business needs change and a better understanding of the system is developed
  - Different viewpoints have different requirements and these are often contradictory
- **Requirement change because:**
  - 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
- **Principal stages consists of:**
  - 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 evolution


Initial understanding of problem
Initial requirements
Changed understanding of prob lem
Changed requirements
Time

Requirement Elicitation and Analysis Techniques

- Interviews/Questionnaire
- Brainstorming Sessions
- Facilitated Application Specification Technique (FAST)
- Use Case Approach
- Ethnography
- Prototyping
- Quality Function Deployment (QFD) [Read yourself]

Use-case Diagram

- A diagram that depicts the interactions between the system and external systems and users.
- It graphically describes who will use the system and in what ways the user expects to interact with the system.
- A use case diagram shows the relationships between actors and their interactions with a system.

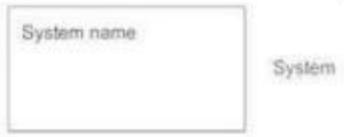
Use Case Symbols

- Use case – subset of the overall system functionality. Represented graphically by a an oval with the name of the use case inside the oval.
- Actor – anything that needs to interact with the system to exchange information. Could be a human, an organization, another information system, an external device.
- Actors communicate by sending and receiving stimuli to and from the system. Each actor has a name

Basic Use Case Diagram Symbols and Notations

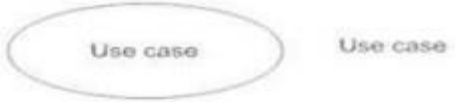
System

Draw your system's boundaries using a rectangle that contains use cases. Place actors outside the system's boundaries.



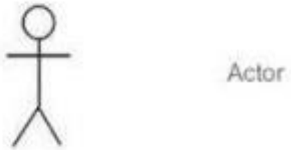
Use Case

Draw use cases using ovals. Label the ovals with verbs that represent the system's functions



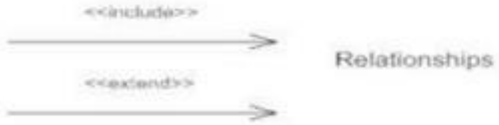
Actors

Actors are the users of a system. Actors interacts with the system by receiving or sending flow of information.

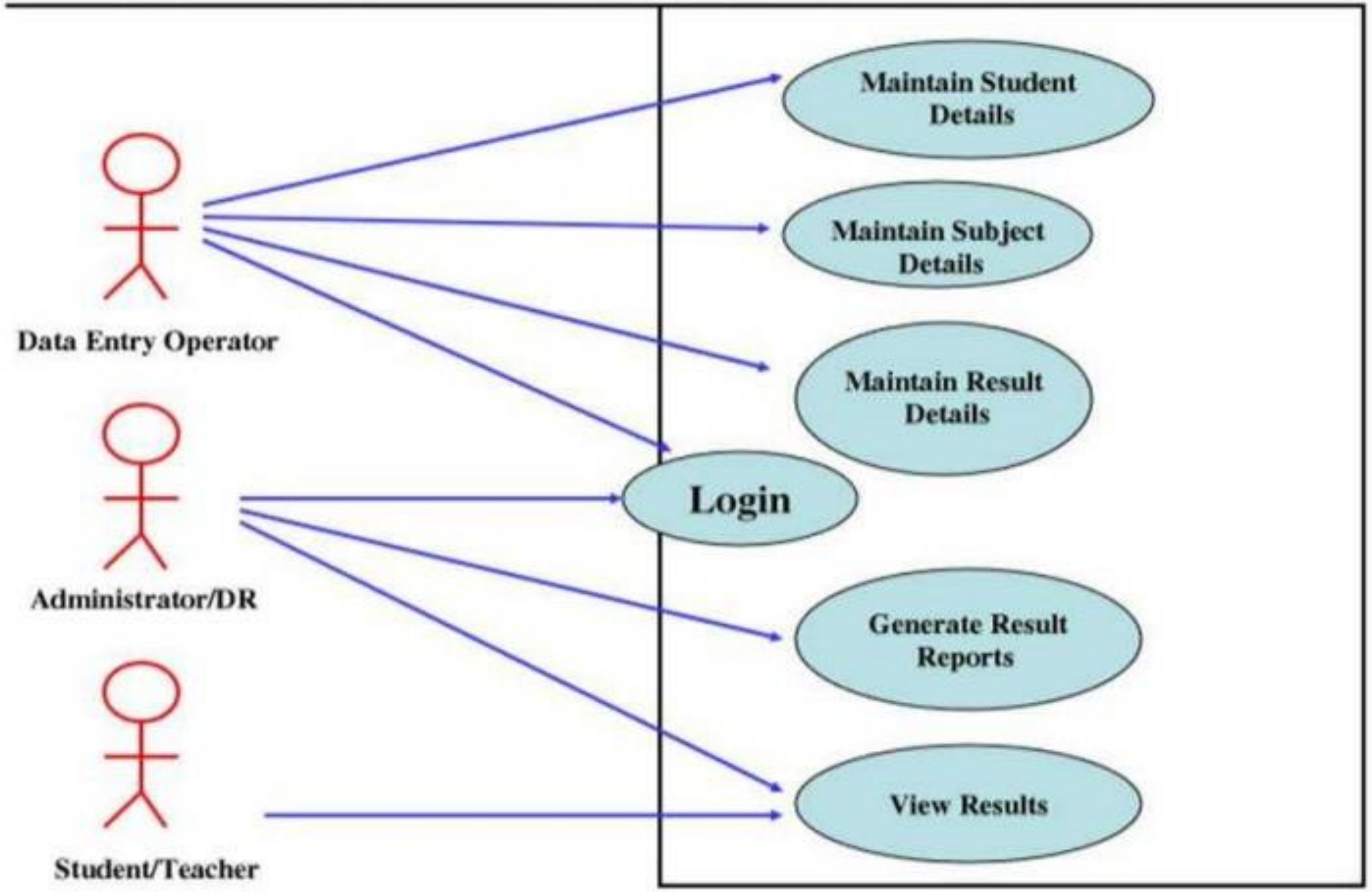


Relationships

Illustrate relationships between an actor and a use case with a simple line. For relationships among use cases, use arrows labeled either "uses" or "extends." A "uses" relationship indicates that one use case is needed by another in order to perform a task. An "extends" relationship indicates alternative options under a certain use case



Use case diagram for Result Management System



Purpose of Use Case Diagram

Use case diagrams are typically developed in the early stage (Analysis phase) of development and people often apply use case modeling for the following purposes:

- Specify the context of a system
- Capture the requirements of a system
- Validate a systems architecture
- Drive implementation and generate test cases
- Developed by analysts together with domain experts

Benefits of Use-Case Modeling

- Provides a tool for capturing functional requirements.
- Assists in decomposing system scope into more manageable pieces.
- Provides a means of communicating with users and other stakeholders concerning system functionality in a language that is easily understood.
- Provides a means of identifying, assigning, tracking, controlling, and management system development activities, especially incremental and iterative development.
- Provides an aid in estimating project scope, effort, and schedule.
- Provides a baseline for testing in terms of defining test plans and test cases.
- Provides a baseline for user help systems and manuals as well as system development documentation.
- Provides a tool for requirements traceability.
- Provides a starting point for the identification of data objects or entities.
- Provides functional specifications for designing user and system interfaces.
- Provides a means of defining database access requirements.
- Provides a framework for driving the system development project

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;
  - A description of the state when the scenario finishes.

Ethnography

- Getting information by observation
- One of the goals is to use the viewpoint of the people within the system.
- The terms ***emic*** and ***etic*** are used.
- An ***etic*** view of the system is the 'outside view' or what the analysts see.
- An ***emic*** view is the 'inside view' or what the system users see.
- The main characteristics of ethnographic studies are:
  - analysts observe or possible even participate in such activities.
  - any interviews are conducted ***in situ,*** possibly as informal discussion rather than formal interview.
  - there is emphasis on examining the interaction between users.
  - there is emphasis on tracing communication links and
  - there is detailed analysis of artifacts.

Interviewing

- In formal or informal interviewing, the RE team puts questions to stakeholders about the system that they use and the system to be developed.
- There are two types of interview
  - Closed interviews where a pre-defined set of questions are answered.
  - Open interviews where there is no pre-defined agenda and a range of issues are explored with stakeholders.

Feasibility studies

- A feasibility study decides whether or not the proposed system is worthwhile.
- A short focused study that checks
  - If the system contributes to organisational objectives;
  - If the system can be engineered using current technology and within budget;
  - If the system can be integrated with other systems that are used.

[Recall SAD for detail ☺ figure, types and steps ☺]

Chapter 3 Ends !