MALLA REDDY UNIVERSITY

R22 - II YEAR / I SEM

(MR22-1CS0146)

OBJECT ORIENTED SOFTWARE ENGINEERING

A. Y. 2023 – 2024

UNIT–II

Software Engineering Principles: Software Requirements: Functional and non-functional requirements, user requirements, system requirements, interface specification, the software requirements document.

Requirements Engineering: Feasibility studies, requirements elicitation and analysis, requirements validation, requirements management

SOFTWARE ENGINEERING PRINCIPLES

SOFTWARE REQUIREMENTS:

Software requirements are necessary

- To introduce the concepts of user and system requirements
- To describe functional and non-functional requirements
- To explain how software requirements may be organized in a requirements document

What is a requirement?

- The requirements for the system are the description of the services provided by the system and itsoperational constraints
- It may range from a high-level abstract statement of a service or of a system constraint to a detailedmathematical 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 thesestatements may be called requirements

Requirements engineering:

- The process of finding out, analysing documenting and checking these services and constraints iscalled requirement engineering.
- The process of establishing the services that the customer requires from a system and the constraintsunder which it operates and is developed.
- The requirements themselves are the descriptions ofthe system services and constraints that aregenerated during the requirements engineering process.

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 organisation's needs. Once a contract has been awarded, the contractor must write a* ***system definition*** *for the client in more detail so that the client understands and can validate what the software will do. Both of these documents may be called the* ***requirements document*** *for the system."*

Types of requirements:

- **User requirements**
    - Statements in natural language plus diagrams ofthe services the system provides and itsoperational 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.

Definitions and specifications:

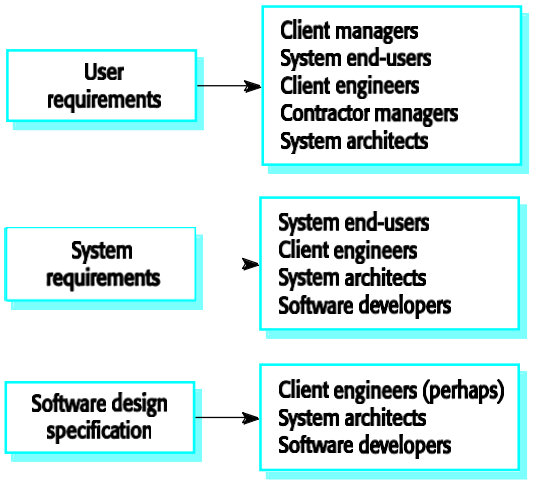
User Requirement Definition:

The software must provide the means of representing and accessing external files created by other tools.

System Requirement specification:

- The user should be provided with facilities to define the type of external files.
- Each external file type may have an associated tool which may be applied to the file.
- Each external file type may be represented as a specific icon on the user's display.
- Facilities should be provided for the icon representing an external file type to be defined bythe user.
- When a user selects an icon representing an external file, the effect of that selection is to apply the tool associated with the type of the external file to the file represented by the selected icon.

Requirements readers:



Functional and non-functional requirements:

Functional requirements

- Statements of services the system should provide how the system should react to particularinputs and how the system should behave in particularsituations.

Non-functional requirements

- Constraints on the services or functions offered by the system such as timing constraints,constraints on the development process, standards, etc.

Domain requirements

- Requirements that come from the application domain of the system and that reflectcharacteristics of that domain.

1. FUNCTIONAL REQUIREMENTS:

- Describe functionality or system services.

- Depend on the type of software, expected users and the type of system where the softwareisused.
- Functional user requirements may be high-level statements of what the system should do butfunctional system requirements should describe the system services in detail.

The functional requirements for **The LIBSYS system:**

- A library system that provides a single interface to a number of databases of articles in differentlibraries.
- Users can search for, download and print these articles for personal study.

Examples of functional requirements

- The user shall be able to search either all ofthe initial set of databases or select a subset from it.
- The system shall provide appropriate viewers for the user to read documents in the document store.
- Every order shall be allocated a unique identifier (ORDER_ID) which the user shall be able to copyto the account's permanent storage area.

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 'appropriate viewers'
  - User intention - special purpose viewer for each different document type;
  - Developer interpretation - Provide a text viewer that shows the contents of the document.

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.

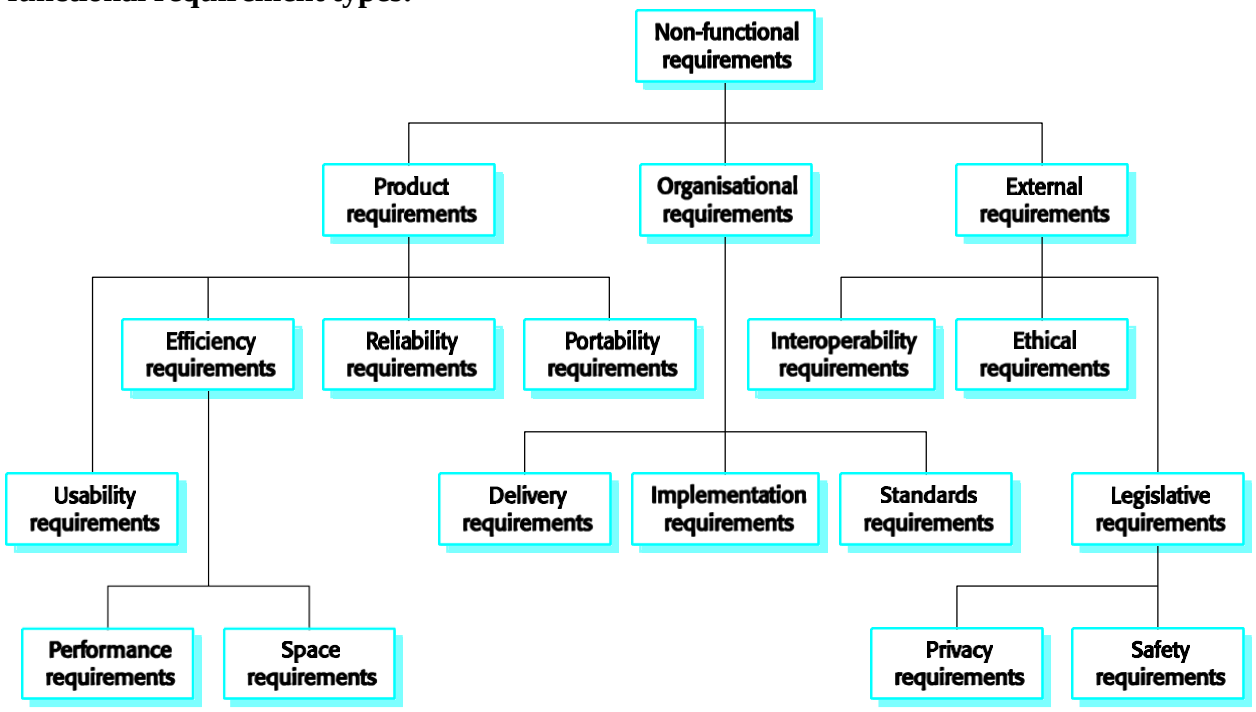
Inpractice, it is impossible to produce a complete and consistent requirements document.

2. NON-FUNCTIONAL REQUIREMENTS

- These define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular CASE system, programminglanguage or development method.

- Non-functional requirements may be more critical than functional requirements. If these are not met,the system is useless.

Non-functional requirement types:



Non-functional requirements:

Product requirements

- Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
- *Eg:*The user interface for LIBSYS shall be implemented as simple HTML without frames orJava applets.

Organizational requirements

- Requirements which are a consequence of organizational policies and procedurese.g.process standards used, implementation requirements, etc.
- *Eg:* The system development process and deliverable documents shall conform to theprocess and deliverables defined in XYZCo-SP-STAN-95.

External requirements

- Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.
- *Eg:* The system shall not disclose any personal information about customers apart from theirname and reference number to the operators of the system.

Goals and requirements:

- Non-functional requirements may be very difficult to state precisely and imprecise requirements maybe difficult to verify.
- Goal

- A general intention of the user such as ease of use.
- The system should be easy to use by experienced controllers and should be organized in such a way that user errors are minimized.

- Verifiable non-functional requirement
  - A statement using some measure that can be objectively tested.
  - Experienced controllers shall be able to use all the system functions after a total of two hours training. After this training, the average number of errors made by experienced users shall not exceed two per day.
- Goals are helpful to developers as they convey the intentions of the system users.

Requirements measures:

Property	Measure
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	M Bytes 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 failureProbability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

Requirements interaction:

- Conflicts between different non-functional requirements are common in complex systems.
- Spacecraft system
  - To minimize weight, the number of separate chips in the system should beminimized.
  - To minimize power consumption, lower power chips should be used.
- However, using low power chips may mean that more chips have to be used. Which is the mostcritical requirement?

A common **problem with non-functional requirements** is that they can be difficult to verify. Users or customers often state these requirements as general goals such as ease of use, the ability of the system to recover from failure or rapid user response. These vague goals cause problems for system developers as theyleave scope for interpretation and subsequent dispute once the system is delivered.

DOMAIN REQUIREMENTS

- Derived from the application domain and describe system characteristics and features that reflect thedomain.
- Domain requirements be new functional requirements, constraints on existing requirements or definespecific computations.
- If domain requirements are not satisfied, the system may be unworkable.

Library system domain requirements:

- There shall be a standard user interface to all databases which shall be based on the Z39.50 standard.
- Because of copyright restrictions, some documents must be deleted immediately on arrival. Dependingon the user's requirements, these documents will either be printed locally on the system server for manually forwarding to the user or routed to a network printer.

Domain requirements problems Understandability

- Requirements are expressed in the language of the application domain;
- This is often not understood by software engineers developing the system.

Implicitness

- Domain specialists understand the area so well that they do not think of makingthe domainrequirements explicit.

USER REQUIREMENTS:

- Should describe functional and non-functional requirements in such a way that they are understandable by system users who don't have detailed technicalknowledge.
- User requirements are defined using natural language, tables and diagrams as these can be understood by all users.

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 bemixed-up.

Requirements amalgamation

- Several different requirements may be expressed together.

Requirement problems

Database requirements includes both conceptual and detailed information

- Describes the concept of a financial accounting system that is to be included in LIBSYS;
- However, it also includes the detail that managers can configure this system - this is unnecessary at this level.

Grid requirement mixes three different kinds of requirement

- Conceptual functional requirement (the need for a grid);
- Non-functional requirement (grid units);
- Non-functional UI requirement (grid switching).
- Structured presentation

Guidelines for writing requirements

- Invent a standard format and use it for all requirements.
- Use language in a consistent way. Use shall for mandatory requirements, should for desirablerequirements.
- Use text highlighting to identify key parts of the requirement.
- Avoid the use of computer jargon.

SYSTEM REQUIREMENTS:

- More detailed specifications of system functions, services and constraints than user requirements.
- They are intended to be a basis for designing the system.
- They may be incorporated into the system contract.
- System requirements may be defined or illustrated using system models

Requirements and design

In principle, requirements should state what the system should do and the design should describehow 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 design may be a domain requirement.

Problems with NL (natural language) specification

Ambiguity

- The readers and writers of the requirement must interpret the same words in the same way.NL is naturally ambiguous so this is very difficult.

Over-flexibility

- The same thing may be said in a number of different ways in the specification.

Lack of modularization.

- NL structures are inadequate to structure system requirements.

Alternatives to NL specification:

Notation	Description
Structured natural language	This approach depends on defining standard forms or templates to express the requirements specification.
Design description languages	This approach uses a language like a programming language but with more abstract features to specify the requirements by defining an operational model of the system. This approach is not now widely used although it can be useful for interface specifications.
Graphical notations	A graphical language, supplemented by text annotations is used to define the functional requirements for the system. An early example of such a graphical language was SADT. Now, use-case descriptions and sequence diagrams are commonly used.
Mathematical specifications	These are notations based on mathematical concepts such as finite-state machines or sets. These unambiguous specifications reduce the arguments between customer and contractor about system functionality. However, most customers don't understand formal specifications and are reluctant to accept it as a system contract.

Structured language specifications

- The freedom of the requirements writer is limited by a predefined template for requirements.
- All requirements are written in a standard way.
- The terminology used in the description may belimited.
- The advantage is that the most of the expressiveness of natural language is maintained but a degreeof uniformity is imposed on the specification.

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.
- Indication of other entities required.
- Pre and post conditions (if appropriate).
- The side effects (if any) of the function.

Tabular specification

- Used to supplement natural language.
- Particularly useful when you have to define a number of possible alternative courses of action.

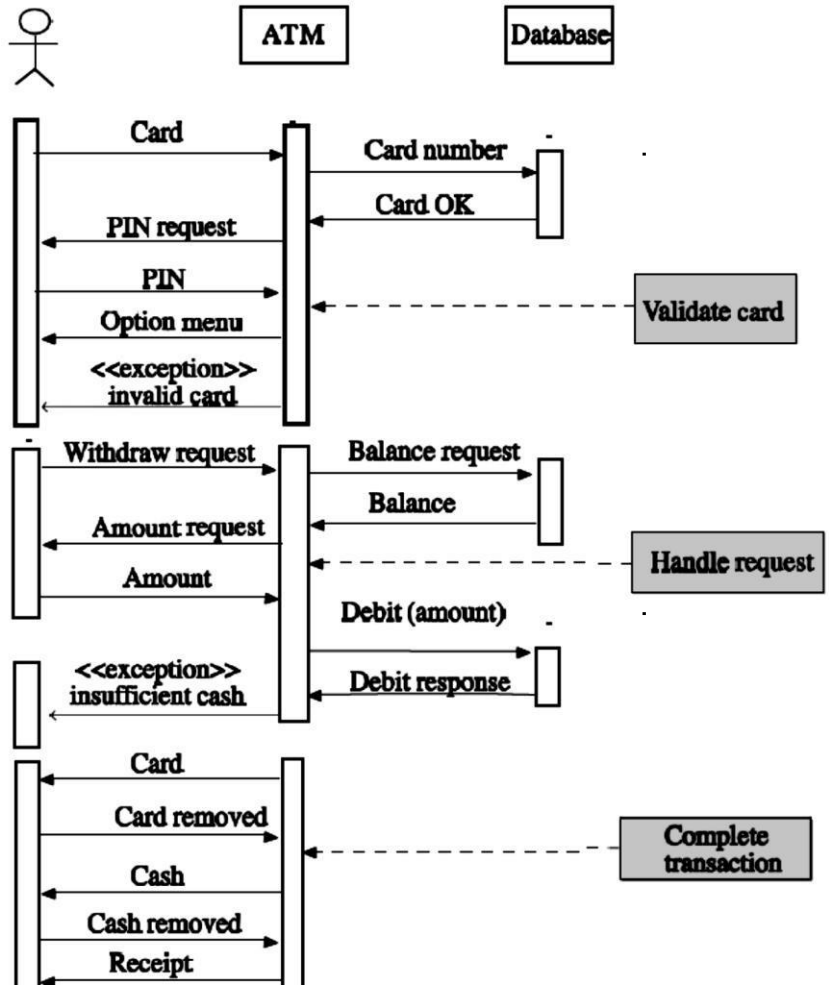
Graphical models

- Graphical models are most useful when you need to show how state changes or where you need todescribe a sequence of actions.

Sequence diagrams

- These show the sequence of events that take place during some user interaction with a system.
- You read them from top to bottom to see the order of the actions that take place.
- Cash withdrawal from an ATM
  - Validate card;
  - Handle request;
    - Complete transaction.

Sequence diagram of ATM withdrawal



System requirement specification using a standard form:

1. Function
2. Description
3. Inputs
4. Source
5. Outputs
6. Destination
7. Action
8. Requires
9. Pre-condition
10. Post-condition
11. Side-effects

When a standard form is used for specifying functional requirements, the following information should beincluded:

1. Description of the function or entity being specified
2. Description of its inputs and where these come from
3. Description of its outputs and where these go to
4. Indication of what other entities are used
5. Description of the action to be taken
6. If a functional approach is used, a pre-condition setting out what must be true before the function iscalled and a post-condition specifying what is true after the function is called
7. Description of the side effects of the operation.

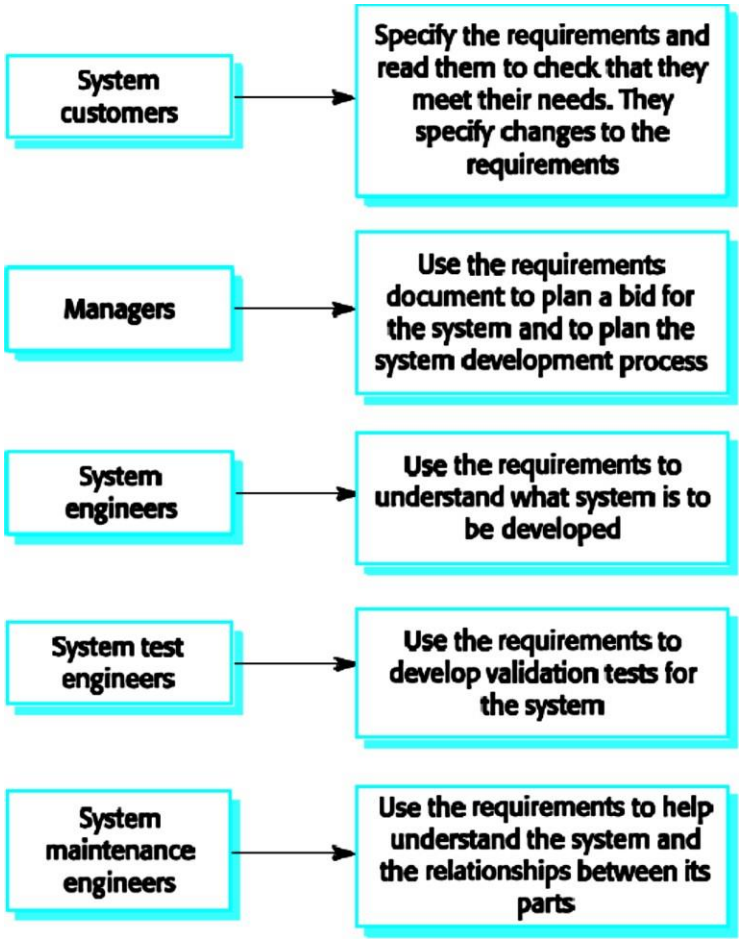
INTERFACE SPECIFICATION:

- Most systems must operate with other systems and the operating interfaces must be specified as part of the requirements.
- Three types of interface may have to be defined
  - ***Procedural interfaces*** where existing programs or sub-systems offer a range of services that are accessed by calling interface procedures. These interfaces are sometimes called ApplicatinProgramming Interfaces (APIs)
  - ***Data structures that are exchanged t***hat are passed from one sub-system to another. Graphical data models are the best notations for this type of description
  - ***Data representations*** that have been established for an existing sub-system
- Formal notations are an effective technique for interface specification.

THE SOFTWARE REQUIREMENTS DOCUMENT:

- The 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 ratherthan HOW it should do it

Users of a requirements document:



IEEE requirements standard defines a generic structure for a requirements document that must be instantiated for each specific system.

1. Introduction.
   i) Purpose of the requirements document
   ii) Scope of the project
   iii) Definitions, acronyms and abbreviations
   iv) References
   v) Overview of the remainder of the document
2. General description.
   i) Product perspective
   ii) Product functions
   iii) User characteristics
   iv) General constraints
   v) Assumptions and dependencies
3. Specific requirements cover functional, non-functional and interface requirements. The requirementsmay document external interfaces, describe system functionality and performance, specify logical database requirements, design constraints, emergent system properties and quality characteristics.
4. Appendices.
5. Index.

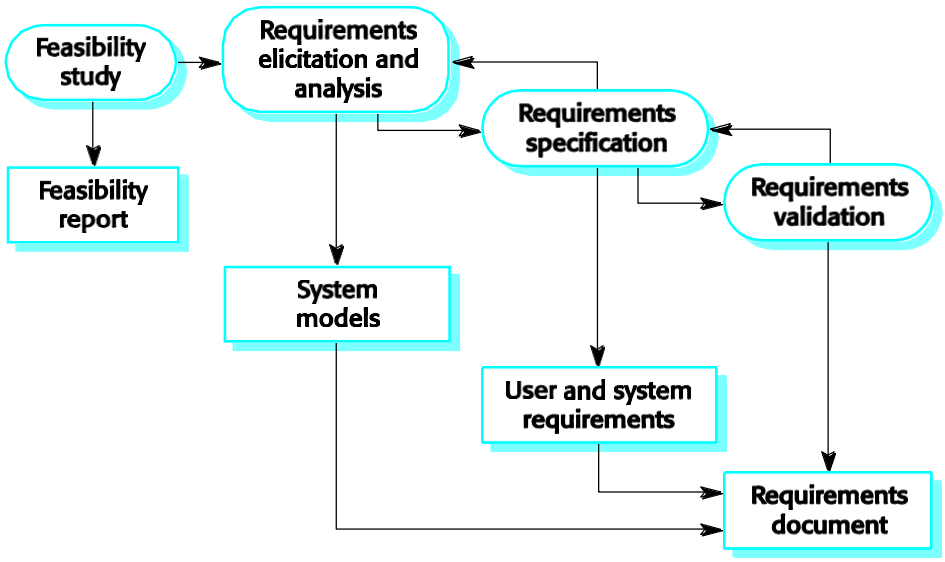
REQUIREMENTS ENGINEERING

Requirement Engineering is the disciplined, process-oriented approach to the development and management of requirements.

The **goal** of requirements engineering process is to create and maintain a system requirements document. The overall process includes four high-level requirement engineering sub-processes. These are concerned with

- ✓ Assessing whether the system is useful to the business (feasibility study)
- ✓ Discovering requirements (elicitation and analysis / Requirement Gathering)
- ✓ Converting these requirements into some standard form(specification)
- ✓ Checking that the requirements actually defines the system that the customer wants(validation). The process of managing the changes in the requirements is called **requirement management**.

The requirements engineering process



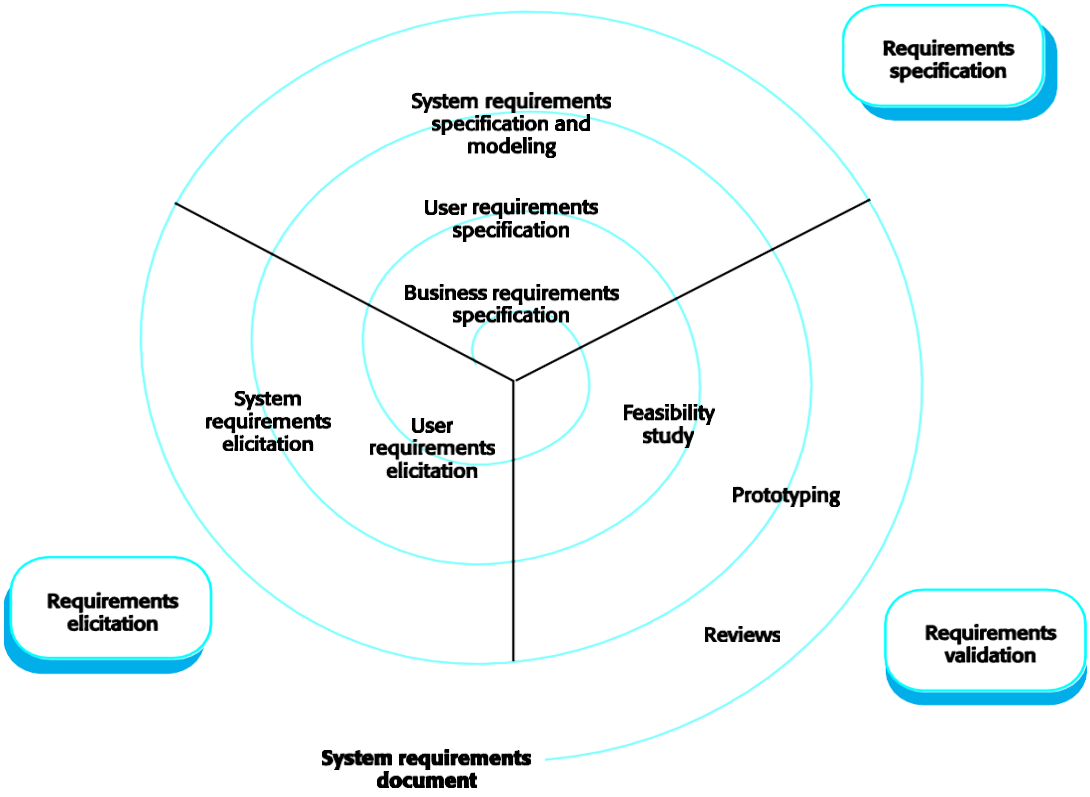
Requirements engineering:

The alternative perspective on the requirements engineering process presents the process as a **three-stage activity** where the activities are organized as an iterative process around a spiral. The amount of time and effort devoted to each activity in iteration depends on the stage of the overall process and the type of system being developed. Early in the process, most effort will be spent on understanding high-level business and non-functional requirements and the user requirements. Later in the process, in the outer rings of the spiral, more effort will be devoted to system requirements engineering and system modeling.

This spiral model accommodates approaches to development in which the requirements are developed to different levels of detail. The number of iterations around the spiral can vary, so the spiral can be exited aftersome or all of the user requirements have been elicited.

Some people consider requirements engineering to be the process of applying a structured analysis method such as object-oriented analysis. This involves analyzing the system and developing a set of graphical systemmodels, such as use-case models, that then serve as a system specification. The set of models describes the behavior of the system and are annotated with additional information describing, for example, its required performance or reliability.

Spiral model of requirements engineering processes



FEASIBILITY STUDIES:

A feasibility study decides whether or not the proposed system is worthwhile. The input to the feasibility study is a set of preliminary business requirements, an outline description of the system and how the system is intended to support business processes. The results of the feasibility study should be a report that recommends whether or not it worth carrying on with the requirements engineering and system developmentprocess.

- A short focused study that checks
  - If the system contributes to organizational 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.

Feasibility study implementation:

- A feasibility study involves information assessment, information collection and report writing.
- Questions for people in the organization
  - What if the system wasn't implemented?
  - What are current process problems?
  - How will the proposed system help?
  - What will be the integration problems?
  - Is new technology needed? What skills?
  - What facilities must be supported by the proposed system?

In a feasibility study, you may consult information sources such as the managers of the departments where the system will be used, software engineers who are familiar with the type of system that is proposed, technology experts and end-users of the system. They should try to complete a feasibility study in two or threeweeks.

Once you have the information, you write the feasibility study report. You should make a recommendation about whether or not the system development should continue. In the report, you may propose changes to the scope, budget and schedule of the system and suggest further high-level requirementsfor the system.

REQUIREMENT ELICITATION AND ANALYSIS:

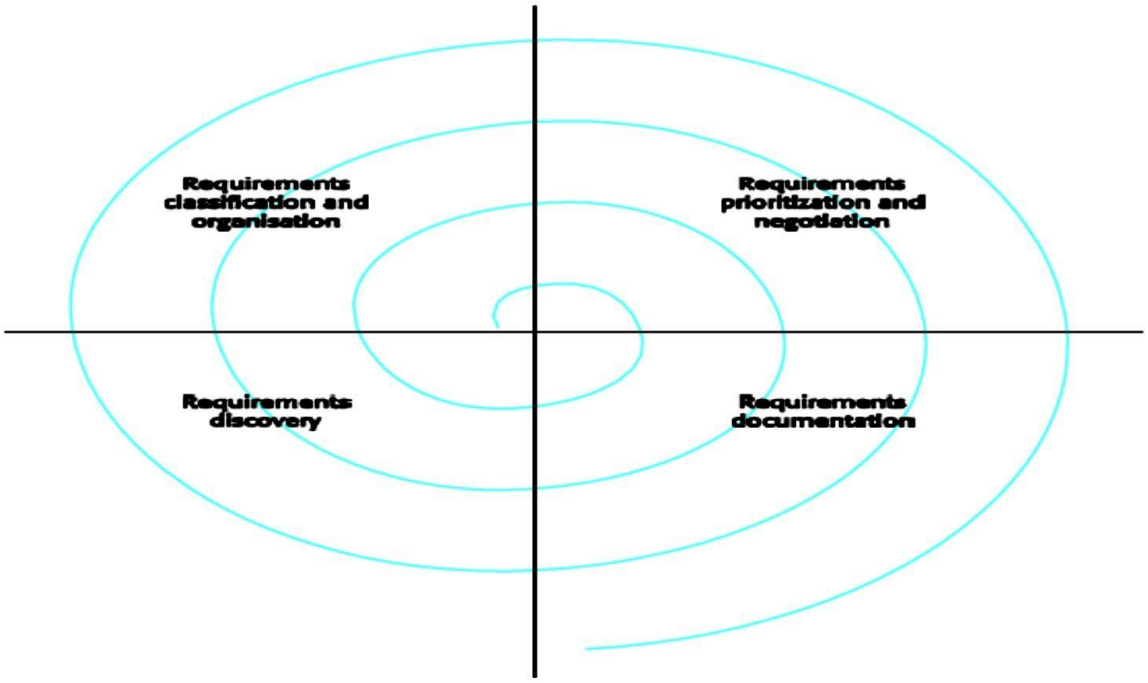
The requirement engineering process is requirements elicitation and analysis.

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

Problems of requirements 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 businessenvironment change.

The requirements spiral



Process activities

1. Requirements discovery
   - Interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage.
2. Requirements classification and organization
   - Groups related requirements and organizes them into coherent clusters.
3. Prioritization and negotiation
   - Prioritizing requirements and resolving requirements conflicts.
4. Requirements documentation
   - Requirements are documented and input into the next round of the spiral.

The process cycle starts with requirements discovery and ends with requirements documentation. The analyst'sunderstanding of the requirements improves with each round of the cycle.

Requirements classification and organization is primarily concerned with identifying overlapping requirements from different stakeholders and grouping related requirements. The most common way of grouping requirements is to use a model of the system architecture to identify subsystems and to associate requirements with each sub-system.

Inevitably, stakeholders have different views on the importance and priority of requirements, and sometimes these view conflict. During the process, you should organize regular stakeholder negotiations so that compromises can be reached.

In the requirement documenting stage, the requirements that have been elicited are documented in such a waythat they can be used to help with further requirements discovery.

1. REQUIREMENTS DISCOVERY:

- Requirement discovery is the process of gathering information about the proposed and existingsystems and distilling the user and system requirements from this information.
- Sources of information include documentation, system stakeholders and the specifications of similarsystems.
- They interact with stakeholders through interview and observation and may use scenarios and prototypes to help with the requirements discovery.
- Stakeholders range from system end-users through managers and external stakeholders such asregulators who certify the acceptability of the system.
- For example, system stakeholder for a bank ATM include
    1. Bank customers
    2. Representatives of other banks
    3. Bank managers
    4. Counter staff
    5. Database administrators
    6. Security managers
    7. Marketing department
    8. Hardware and software maintenance engineers
    9. Banking regulators

Requirements sources (stakeholders, domain, systems) can all be represented as system viewpoints, whereeach viewpoints, where each viewpoint presents a sub-set of the requirements for the system.

Viewpoints:

- Viewpoints are a way of structuring the requirements to represent the perspectives ofdifferentstakeholders. Stakeholders may be classified under different viewpoints.
- This multi-perspective analysis is important as there is no single correct way to analyses systemrequirements.

Types of viewpoint:

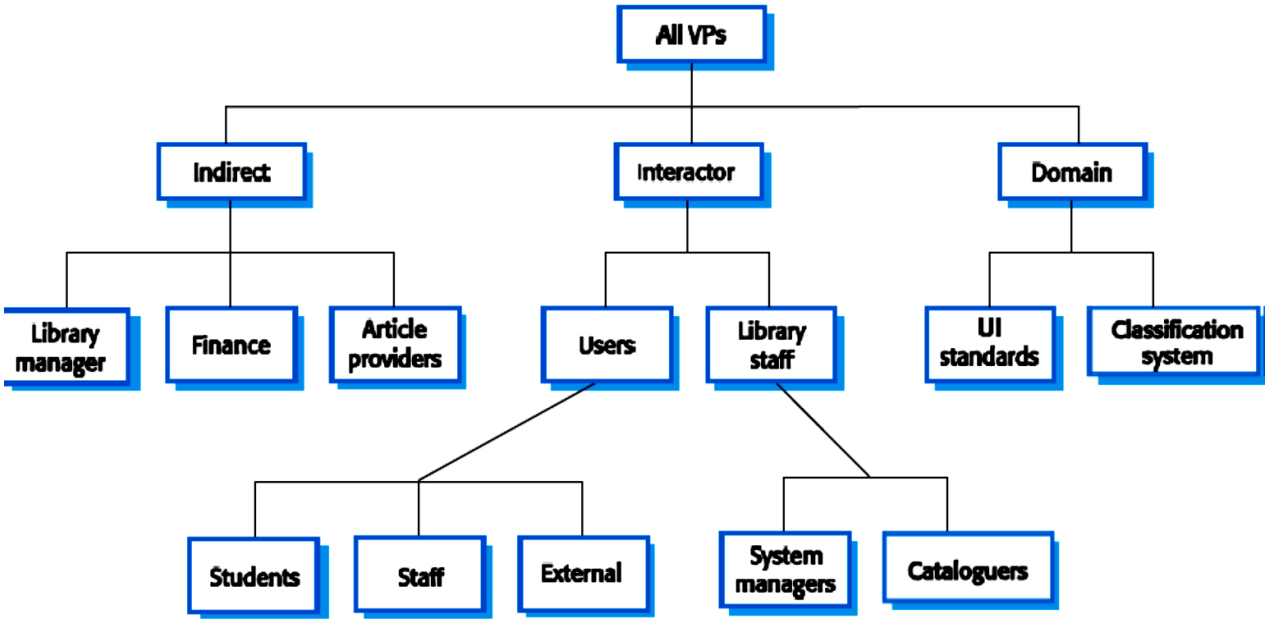
1. **Interactor viewpoints**
    - People or other systems that interact directly with the system. These viewpoints provide detailed system requirements covering the system features and interfaces. In an ATM, the customer's and the account database are interactor VPs.
2. **Indirect viewpoints**
    - Stakeholders who do not use the system themselves but who influence the requirements. These viewpoints are more likely to provide higher-level organisation requirements and constraints. In an ATM, management and security staff are indirectviewpoints.
3. **Domain viewpoints**
    - Domain characteristics and constraints that influence the requirements. These viewpoints normally provide domain constraints that apply to the system. In an ATM, an example wouldbe standards for inter-bank communications.

Typically, these viewpoints provide different types of requirements.

Viewpoint identification:

- Identify viewpoints using
  - Providers and receivers of system services;
  - Systems that interact directly with the system being specified;
  - Regulations and standards;
  - Sources of business and non-functional requirements.
  - Engineers who have to develop and maintain the system;
  - Marketing and other business viewpoints.

LIBSYS viewpoint hierarchy



Interviewing

In formal or informal interviewing, the RE team puts questions to stakeholders about the system that theyuse 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 ofissues are explored withstakeholders.

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 mightinteract 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 itisn't worth articulating.

Effective interviewers:

- Interviewers should be open-minded, willing to listen to stakeholders and should not have pre-conceived ideas about the requirements.
- They should prompt the interviewee with a question or a proposal and should not simplyexpect themto respond to a question such as ‘what do you want’.

Scenarios:

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

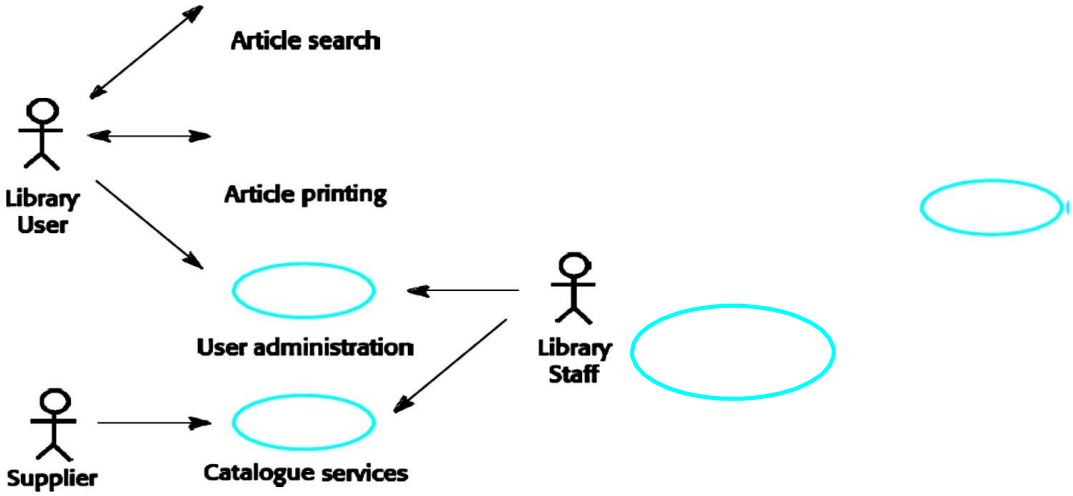
- They should include
  - A description of the starting situation;
  - A description of the normal flow of events;
  - A description of what can go wrong;
  - Information about other concurrent activities;
  - A description of the state when the scenario finishes.

Use cases

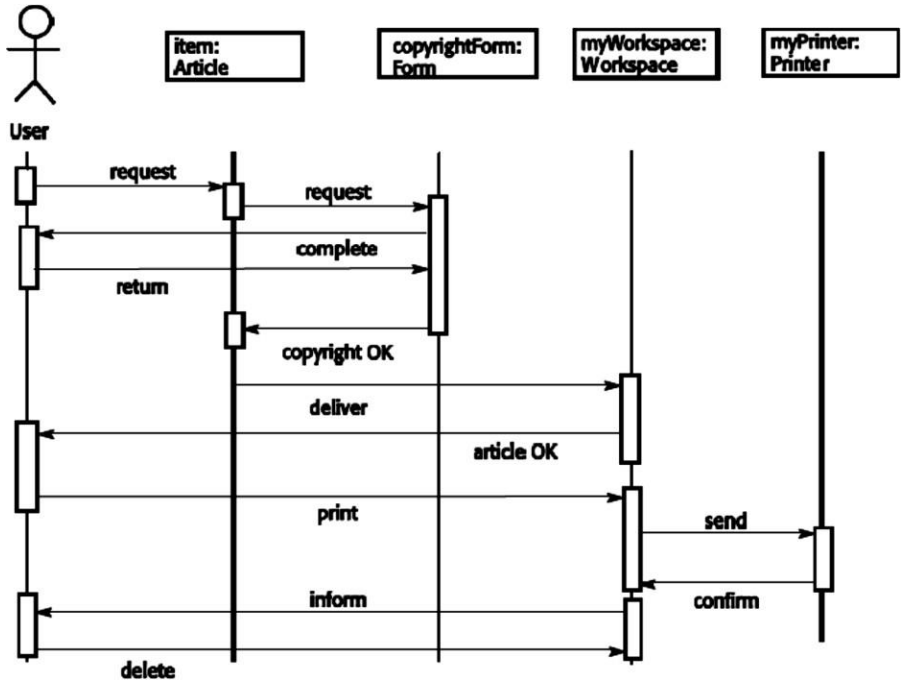
- Use-cases are a scenario based technique in the UML which identify the actors in an interaction andwhich describe the interaction itself.
- A set of use cases should describe all possible interactions with the system.
- Sequence diagrams may be used to add detail to use-cases by showing the sequence ofeventprocessing in the system.
-

Article printing use-case:

LIBSYS use cases:



Article printing sequence:



Social and organizational factors

- Software systems are used in a social and organisational context. This can influence or even dominatethe system requirements.
- Social and organisational factors are not a single viewpoint but are influences on allviewpoints.
- Good analysts must be sensitive to these factors but currently no systematic way to tackle theiranalysis.

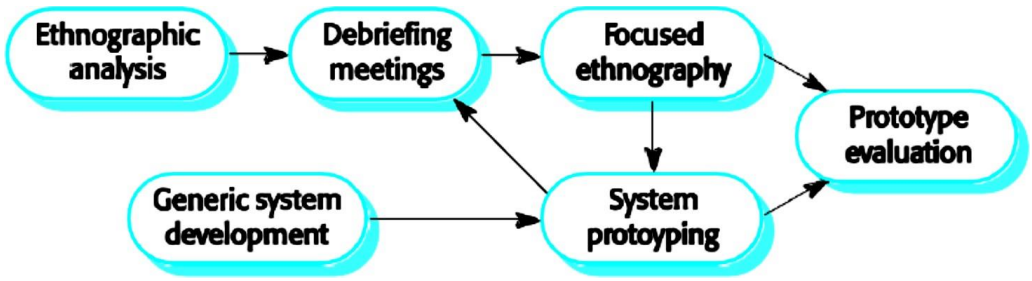
2. ETHNOGRAPHY:

- A social scientists spends a considerable time observing and analyzing how people actuallywork.
- People do not have to explain or articulate their work.
- Social and organisational factors of importance may be observed.
- Ethnographic studies have shown that work is usually richer and more complex than suggested bysimple 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 historicalbasis 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 Iwhichprocess definitions suggest that they ought to work.
- Requirements that are derived from cooperation and awareness of other people's activities.

REQUIREMENTS VALIDATION:

- Concerned with demonstrating that the requirements define the system that the customer really wants.
- Requirements error costs are high so validation is very important
  - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing animplementation 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. Covered in Chapter 17.
- Test-case generation
  - Developing tests for requirements to check testability.

Requirements reviews:

- Regular reviews should be held while the requirements definition is being formulated.
- Both client and contractor staff should be involved in reviews.
- Reviews may be formal (with completed documents) or informal. Good communications betweendevelopers, customers and users can resolve problems at an early stage.

Review checks:

- ***Verifiability*:** Is the requirement realistically testable?
- ***Comprehensibility*:** Is the requirement properlyunderstood?
- ***Traceability*:** Is the origin of the requirement clearlystated?
- ***Adaptability*:** Can the requirement be changed without a large impact on other requirements?

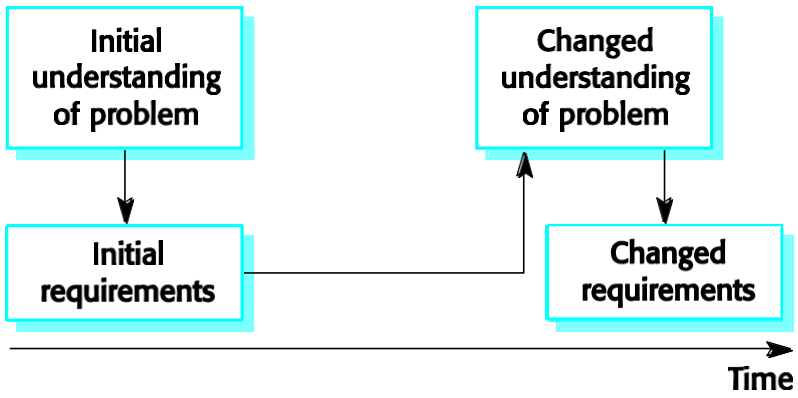
REQUIREMENTS MANAGEMENT:

- Requirements management is the process of managing changing requirements during therequirements 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.

Requirements change

- The priority of requirements from different viewpoints changes during the developmentprocess.
- System customers may specify requirements from a business perspective that conflict withend-userrequirements.
- The business and technical environment of the system changes during its development.

Requirements evolution:



1. Enduring and volatile requirements:

- ***Enduring requirements***: Stable requirements derived from the core activity of the customerorganisation. 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

Requirements classification:

Requirement Type	Description
Mutable requirements	Requirements that change because of changes to the environment in which the organisation is operating. For example, in hospital systems, the funding of patient care may change and thus require different treatment information to be collected.
Emergent requirements	Requirements that emerge as the customer's understanding of the system develops during the system development. The design process may reveal new emergent requirements.
Consequential requirements	Requirements that result from the introduction of the computer system. Introducing the computer system may change the organisations processes and open up new ways of working which generate new system requirements
Compatibility requirements	Requirements that depend on the particular systems or business processes within an organisation. As these change, the compatibility requirements on the commissioned or delivered system may also have to evolve.

2. 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 analysing 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:

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;

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 bedefined andinformation flow between these stages partially automated.
- Traceability management
  - Automated retrieval of the links between requirements.

3. 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 reflectchange.

Change management: