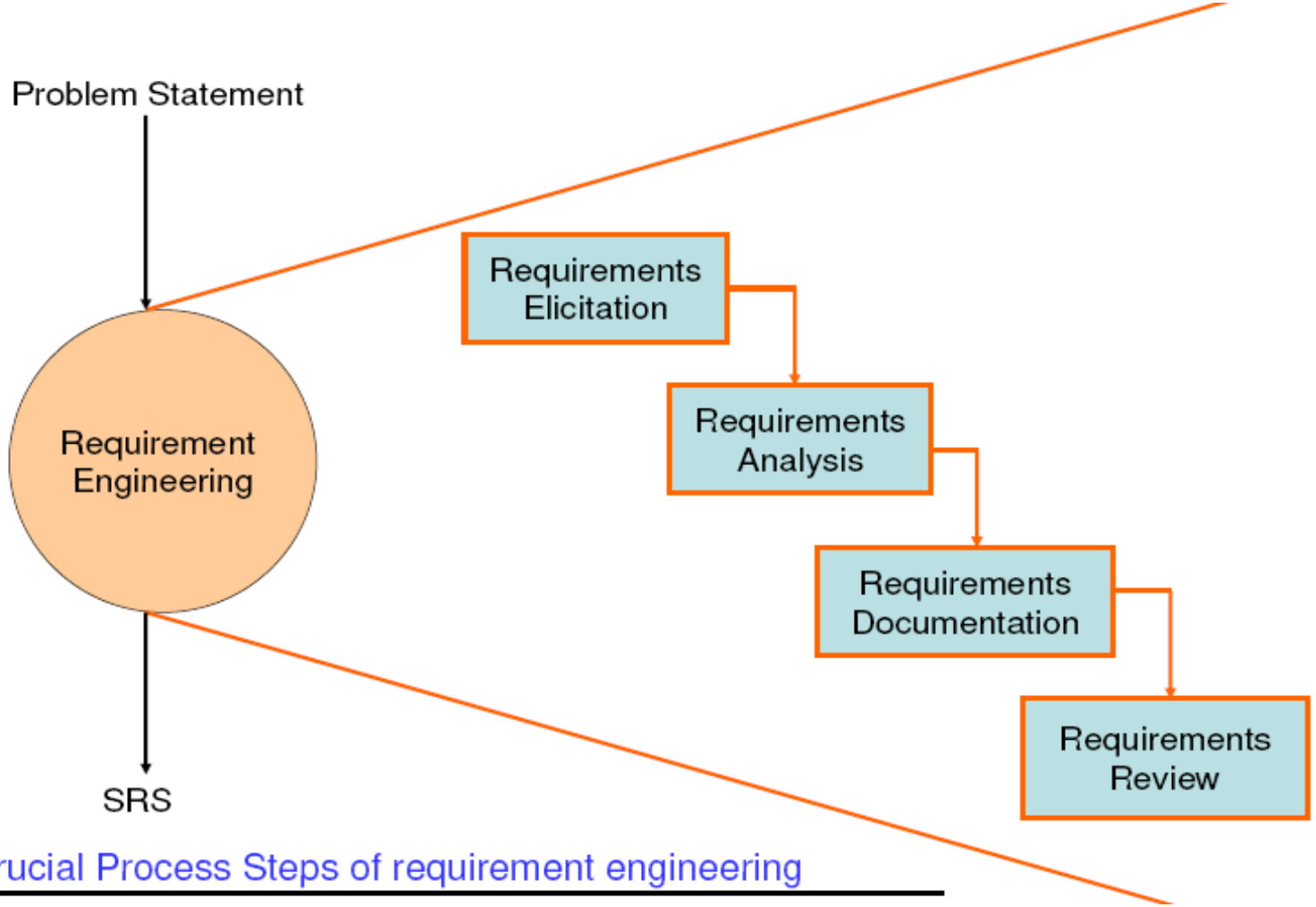
Requirement Engineering (CO2)

- It is the disciplined application of proven principles, methods, tools, and notations to describe a proposed system's intended behavior and its associated constraints.
- It describe the "What" of a system, not the "How".
- It's input is problem statement prepared by the customer.
- It produces one large document(SRS),written in natural language, contain the description of what the system will do with out describing how it will do.
- It's process consists four steps:
  1. Requirement Elicitation.
  2. Requirement Analysis.
  3. Requirement Documentation.
  4. Requirement Review.

Requirement Engineering (CO2)



Crucial Process Steps of requirement engineering

Case Study

A University wish to develop a software system for the student result management of its M.Tech. Programe. A problem statement is to be prepared for the software development company. The problem statement may give an overview of the existing system and broad expectations from the new software system.

Problem statement(prepared by Exam. Division of Univ.)

- Univ. conduct 4-semeste M.Tech. program. student are offered four theory and two practical paper during I,II and IIIrd semester.
- In IV sem. Students have to give a seminar and submit a dissertation on topic area of their interest.
- Evaluation of each theory subject is done out of 150 marks . 100 for univ. conduct exam and 50 for sessional exam, attendance and student assignment.
- Evaluation of practical exam is done out of 50. 25 for univ. exam and 25 for internal in which student prepared lab record, viva, attendance.

Case Study

- Marks of IV sem. Dissertation is 450. 200 if for internal and 250 for external
- If total marks in each subject of a student is 50 % then Student is considered pass other wise Student is failed .
- At any time the latest information about subjects being offered in various semesters and their credit points can be obtained from univ. website.

- It is required to develop a system that manage information about subject offered in various sem. Students enrolled in various sem. Marks obtained by students in different sem.
- The system should also have the ability to generate printable mark-sheets for each student. Semester-wise detailed mark-lists and student performance reports also need to be generated.

Requirement Engineering Process (CO2)

1. Requirement Elicitation
   - It is known as requirement gathering.
   - Req. is identified with the help of customer and existing system process.
2. Requirement Analysis
   - it is start with req. elicitation.
   - It perform to identify inconsistencies, defects, etc.
3. Requirement Documentation
   - It is the end product of 1 and 2. known as SRS.
   - Doc. Provides the foundation of s/w design.
   - SRS may act as a contract between developer and customer.
4. Requirement Review
   - It is carried out to improve the quality of SRS.
   - It may also be called as requirement verification.

Type of Requirements

Stakeholder : Anyone who should have some direct or indirect influence on the system requirements. i.e. end user, developer, tester, coder etc.

- Known Requirements :
  - something a stakeholder believes to be implemented.
- Unknown Requirement :
  - forgotten by the stakeholder because they are not needed right now or needed only by another stakeholder.
- Undreamed Requirement :
  - stakeholder may not be able to think of new requirements due to limited domain knowledge.

Type of Requirement

Known, unknown, undreamed requirements may be functional or non-functional.

- Functional Requirements
  - it is related to the expectations from the intended s/w.
  - It describe what the s/w has to do.
  - Sometimes it may also specify what the s/w should not do.
- Non-Functional Requirements
  - It is mostly quality requirements that stipulate how well the s/w does what it has to do.
  - For **user** includes specification of performance, availability, usability and flexibility.
  - For **developers** are maintainability, portability and testability.

User and System Requirements

- User Requirements
  - User requirement are written for the users and include functional and non functional requirement.

- System Requirements
  - System requirement are derived from user requirement.
  - The user system requirements are the parts of software requirement and specification (SRS) document.

Requirement Elicitation (CO2)

- It is most important goal of RE to find out what users really need.
- It is ability that helps to understand the problem to be solved.
- It is conduct by asking que., writing down answers, asking other que., etc.
- Developers and users have different mind set, expertise and vocabularies so there are chances of conflicts that may led to inconsistencies, misunderstanding and omission of requirements .
- It requires the collaboration of several groups of participants who have different background.

Requirement Elicitation Methods

1. Interviews.
2. Brainstorming Sessions.
3. Facilitated Application Specification Technique(FAST)
4. Quality Function Deployment.
5. The use Case Approach.

Interviews

After receiving problem statement from customer, the first step to arrange a meeting between customer and specialized developer(requirement engineer**)**

Interviews are of two types

I. Open ended: no preset agenda.

II. Structured: a proper questionnaire is designed for interview

Brainstorming Sessions

- It is a group discussion technique that may lead to a lot of new ideas quickly and help to promote creative thinking.
- This technique may carried out with specialized groups like actual users, middle level managers etc, or with total stakeholders.
- To handle any undesirable situations(conflicts) a highly trained facilitator may be required.
- White boards, overhead transparencies or a computer projection system can be used to make it visible to every participant.
- After the session a detailed report will be prepared as facilitator will reviews the report.
- Finally, a document will be prepared which will have list requirements and their priority, if possible.

Facilitated Application Specification Technique(FAST)

- Similar to brainstorming sessions.
- Team oriented approach
- Creation of joint team of customers and developers.

Guidelines

- Arrange a meeting at a neutral site.
- Establish rules for preparation and participation.
- prepare Informal agenda to encourage free flow of ideas.
- Appoint a facilitator to control the meeting.
- Prepare definition mechanism board, worksheets, wall stickier.
- Participants should not criticize or debate.

FAST session Preparations

Each attendee is asked to make a list of objects that are:

- 1. Part of environment that surrounds the system.
- 2. Produced by the system.
- 3. Used by the system.
- A. List of constraints
- B. Functions
- C. Performance criteria

FAST session Preparations

Activities of FAST session

- Every participant presents his/her list of objects, services, constraints and performance for discussion.
- Combine list for each topic by elimination redundant entries and adding new ideas.
- By Discussion, consensus lists are finalized by the facilitator.
- Sub teams develop mini specifications for one or more entries.
- Presentations of mini-specifications, new objects, services, constraints, performance requirements are add to original list.
- Each attendee prepare a list of Validation criteria, a consensus list of validation criteria is then created.
- A sub team write the complete draft specifications using all inputs from FAST meeting.

Quality Function Deployment

- Focus on what is valuable to the customer and deploy these values through the SE process.
- Three types of requirements are identified:
  - Normal requirements: objective and goals of proposed s/w which discussed with the customer.
  - Expected requirements: implicitly req. in s/w that customer not state them explicitly. Eg. Authentication, warning features
  - Exciting requirements: feature go beyond the customer, if they present prove very satisfying. Eg. Unauthorized access notification
- QFD steps:
  - 1. Identify stakeholders
  - 2. List out requirements
  - 3. Degree of importance to each requirement

QFD…

- 5 Points : V. Important
- 4 Points : Important
- 3 Points : Not Important but nice to have
- 2 Points : Not important
- 1 Points : Unrealistic, required further, exploration

- Final list of requirements categorize like:
  - (i) It is possible to achieve
  - (ii) It should be deferred & Why
  - (iii) It is impossible and should be dropped from consideration
- First Category requirements will be implemented as per priority assigned with every requirement.

Use Case Approach

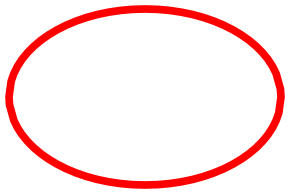
Components of Use Case approach

- Use Cases : it is initiated by a user with a particular goal in mind, and completes successfully when that goal is satisfied.
  - Use Cases are focus on "what" the system is , not "how" the system will be designed.
  - It describes the sequence of interactions between actors and the system necessary to deliver the services that satisfies the goal.
- Actor: An actor or external agent, lies outside the system model, but interacts with it in some way. Actor are Person, machine, information System, etc
- Association : it represent the relationship between actor and useca.se

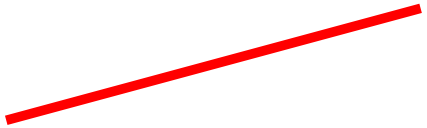
Use case Diagram

-- represents what happens when actor interacts with a system.
-- captures functional aspect of the system.

Actor



Use Case



Relationship between actors and use case and/or between the use cases.

-- Actors appear outside the rectangle.
--Use cases within rectangle providing functionality.
--Relationship association is a solid line between actor & use cases.

Use Case Template

Introduction: Describe a quick background of the use case.

Actors : List the actors that interact and participate in the use cases.

Pre Conditions : Pre conditions that need to be satisfied for the use case to perform.

Post Conditions : Define the different states in which we expect the system to be in, after the use case executes.

Use Case Template

Flow of events

Basic Flow : List the primary events that will occur when this use case is executed.

Alternative Flows: Any Subsidiary events that can occur in the use case should be separately listed. List each such event as an alternative flow. A use case can have many alternative flows as required.

Special Requirements : Business rules will be used for writing test cases. Both success and failures scenarios should be described.

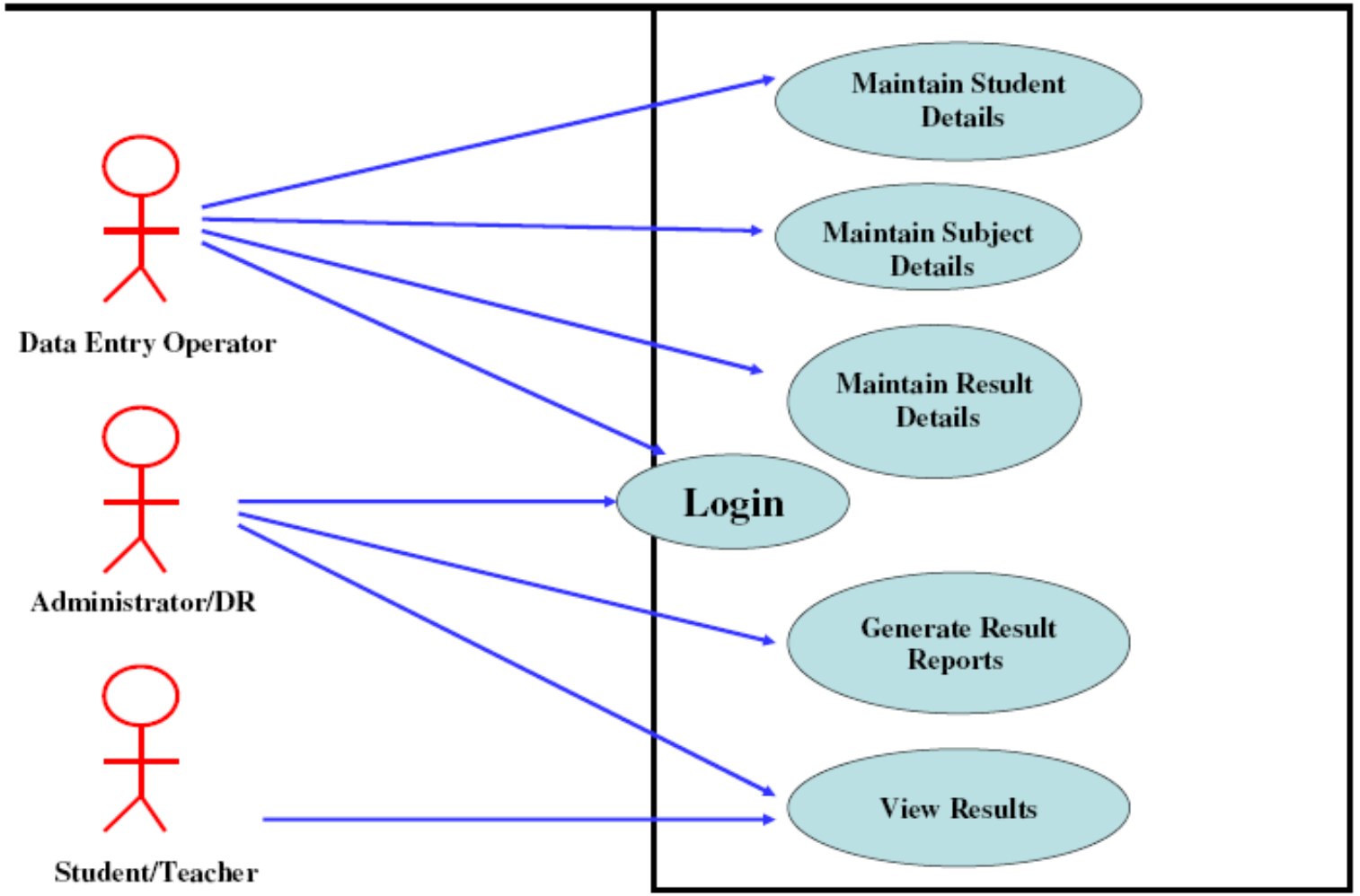
Use Case relationships : For Complex systems Listing the relationships between use cases also provides a mechanism for traceability

Use Case Template

1. Identify all users.
2. Create a user profile for each category of users including all roles of the users play that are relevant to the system.
3. Create a use case for each goal, following the use case template maintain the abstraction throughout the use case.
4. Structure the use case.
5. Review and validate with users.

Cont..

Use case diagram for Result Management System



Login(Template)

Login

- 1.1 Introduction : This use case describes how a user logs into the Result Management System.
- 1.2 Actors : (i) Data Entry Operator
  (ii) Administrator/Deputy Registrar
- 1.3 Pre Conditions : None
- 1.4 Post Conditions :

  If the use case is successful, the actor is logged into the system. If not, the system state is unchanged.

Login(Template)

1.5 Basic Flow : This use case starts when the actor wishes to login to the Result Management system.

- (i) System requests that the actor enter his/her name and password.
- (ii) The actor enters his/her name & password.
- (iii) System validates name & password, and if finds correct allow the actor to logs into the system.

Login(Template)

- 1.6 Alternate Flows
  - 1.6.1 Invalid name & password

If in the basic flow, the actor enters an invalid name and/or password, the system displays an error message. The actor can choose to either return to the beginning of the basic flow or cancel the login, at that point, the use case ends.

1.7 Special Requirements: None

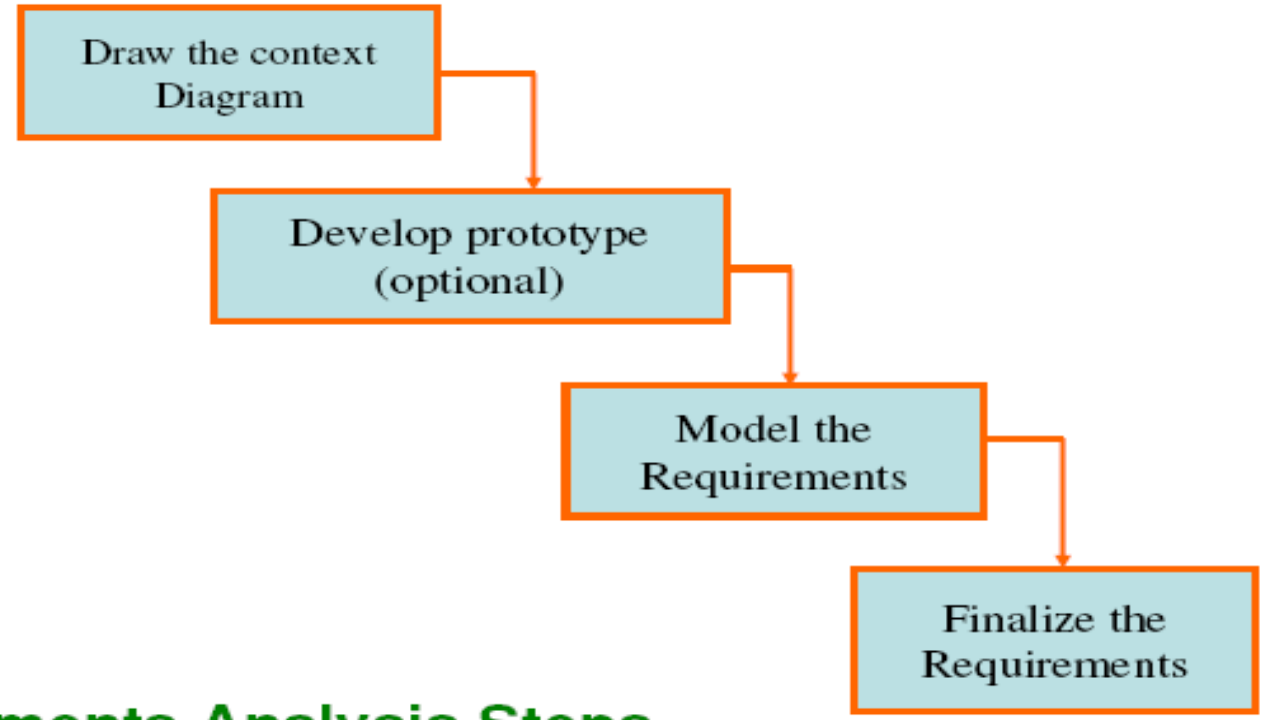
1.8 Use case Relationships: None

Requirement Analysis (CO2)

- We analyze, refine and scrutinize gathered requirements to make consistent & unambiguous requirements.
- Activity reviews all the requirement and may Provide graphical view of the entire system.
- May also interact with customer to resolve their confusion and find priority of requirements.

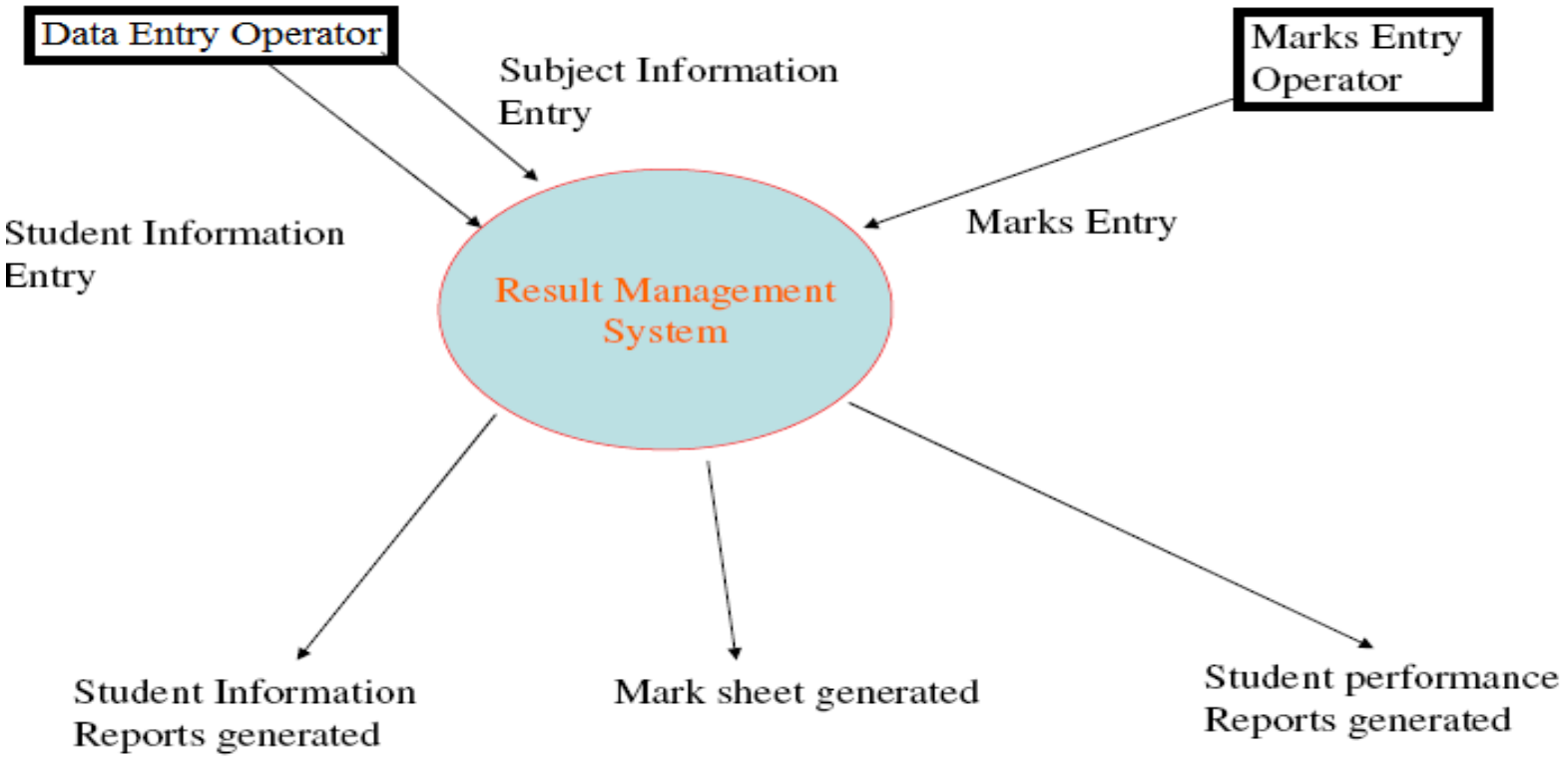
Steps of Requirement Analysis

Steps



Requirements Analysis Steps

Context Diagram (CO2)



Development of Prototype(Optional)

- It is effective way to find out what the customer really want.
- We take customer feedback to continuously modify the prototype until he/she is satisfied.
- It should be built quickly and at a relatively low cost.
- Due to its limitations and would not be acceptable in final system it is an optional activity.
- Many organization are developing prototypes for better understanding before the finalization of SRS.

Model the Requirement

- This process consists of various graphical representation of functions, data entities, external entities and relationship between them.
- Graphical view help us to find incorrect, missing, and inconsistent requirements.
- Such models includes
  - Data flow diagram
  - Entity Relationship Diagram(ERD)
  - Data Dictionary
  - Decision Table, etc.

Data flow diagram

- A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

Levels in Data Flow Diagrams (DFD)

- primarily three levels in the data flow diagram, which are: 0-level DFD, 1-level DFD, and 2-level DFD.

Level-0 DFD

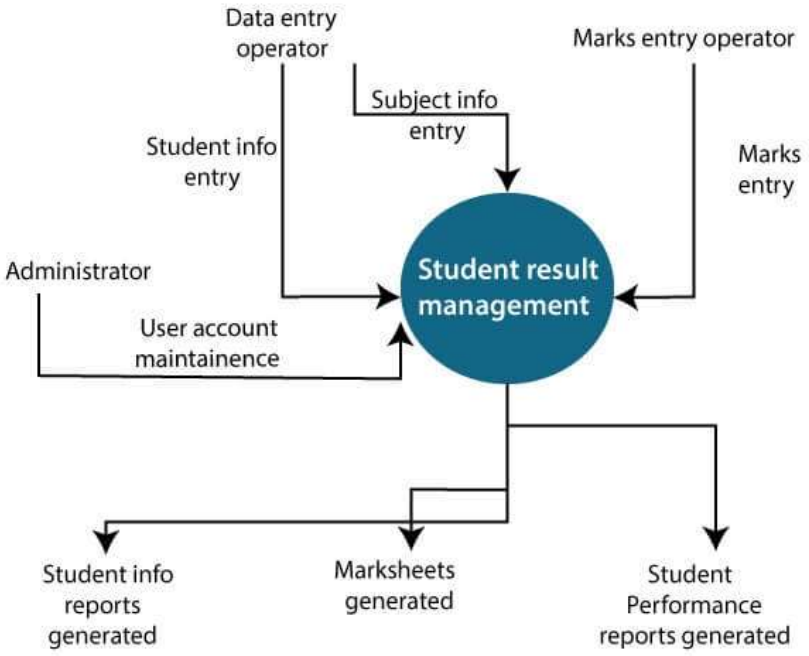


Fig: Level-0 DFD of result management system

- Level-0 DFD: also called context diagram of the result management system is shown in fig. As the bubbles are decomposed into less and less abstract bubbles.

1-level DFD

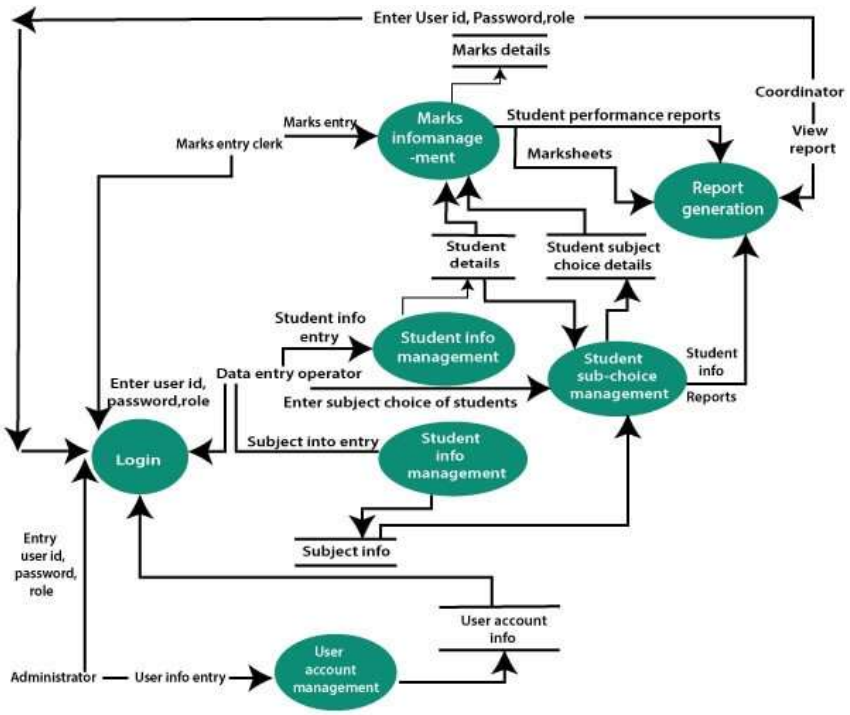
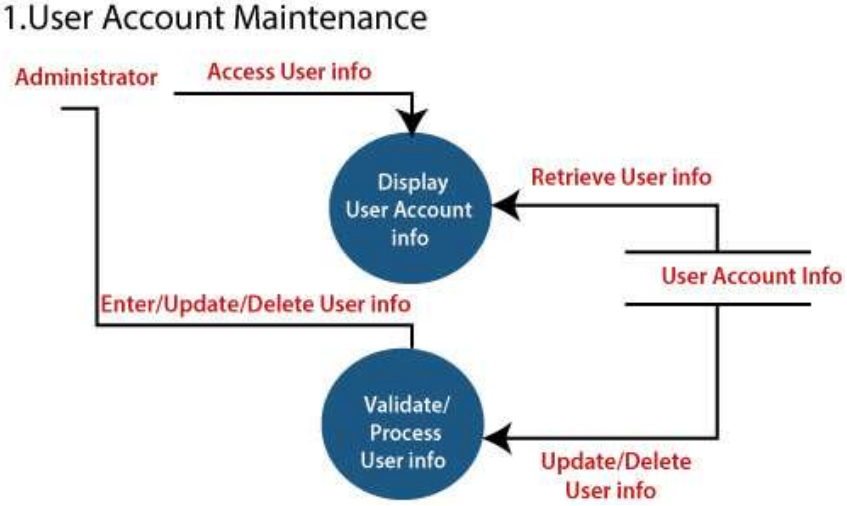
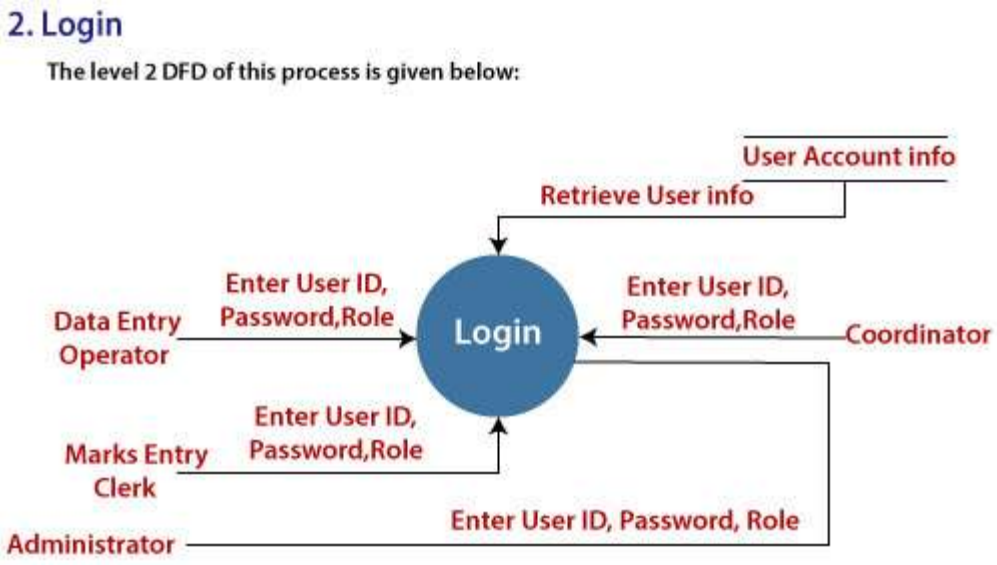


Fig: Level-1 DFD of result management system

- In 1-level DFD, a context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main objectives of the system and breakdown the high-level process of 0-level DFD into subprocesses.

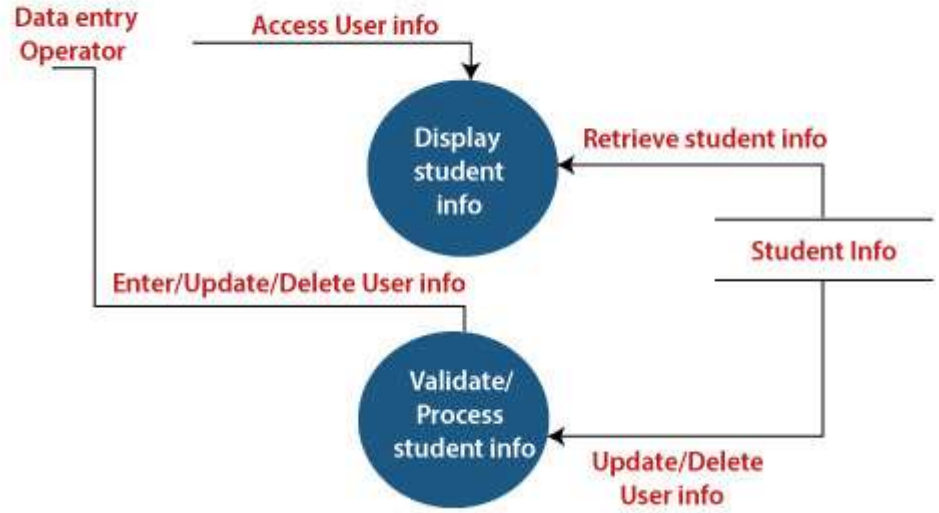
2-Level DFD





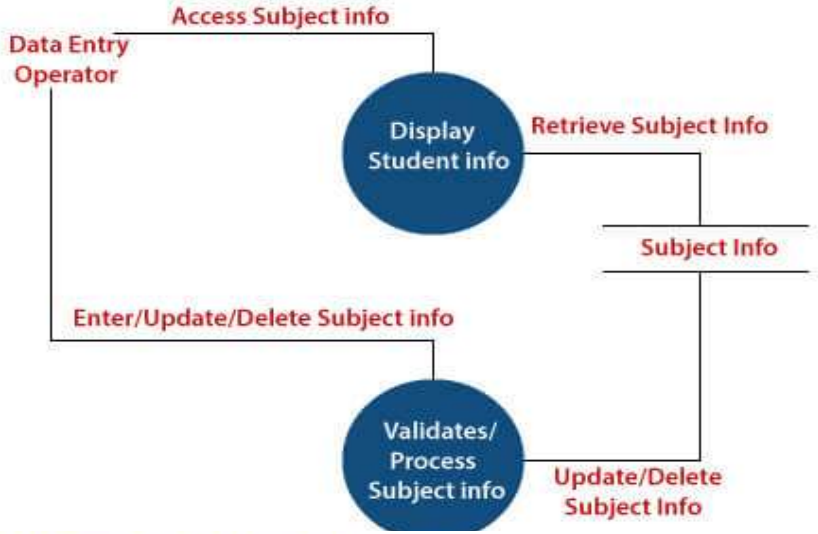
- 2-level DFD goes one process deeper into parts of 1-level DFD. It can be used to project or record the specific/necessary detail about the system's functioning.

3. Student Information Management



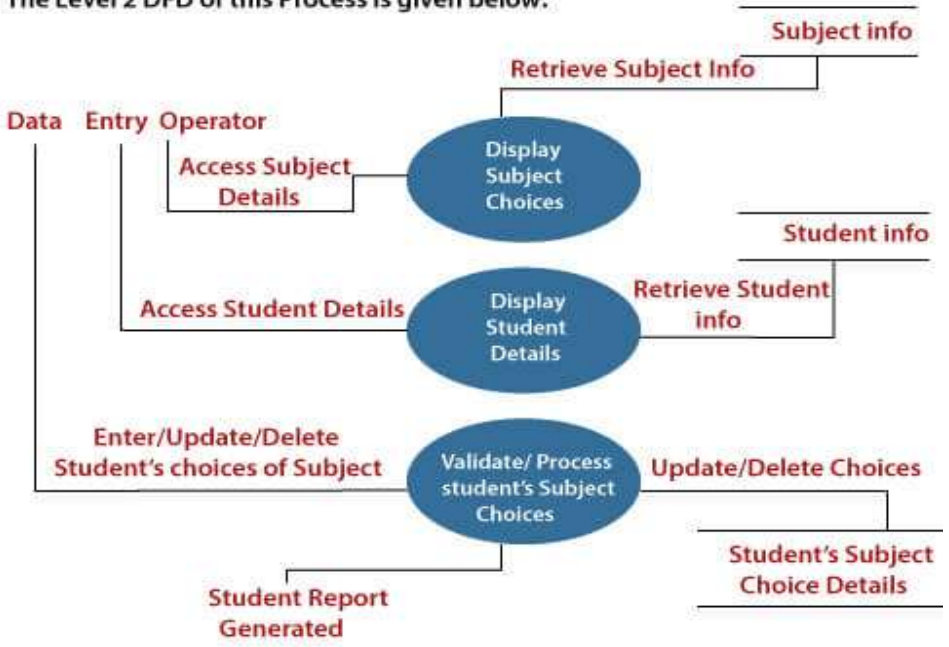
4. Subject Information Management

The level 2 DFD of this process is given below:



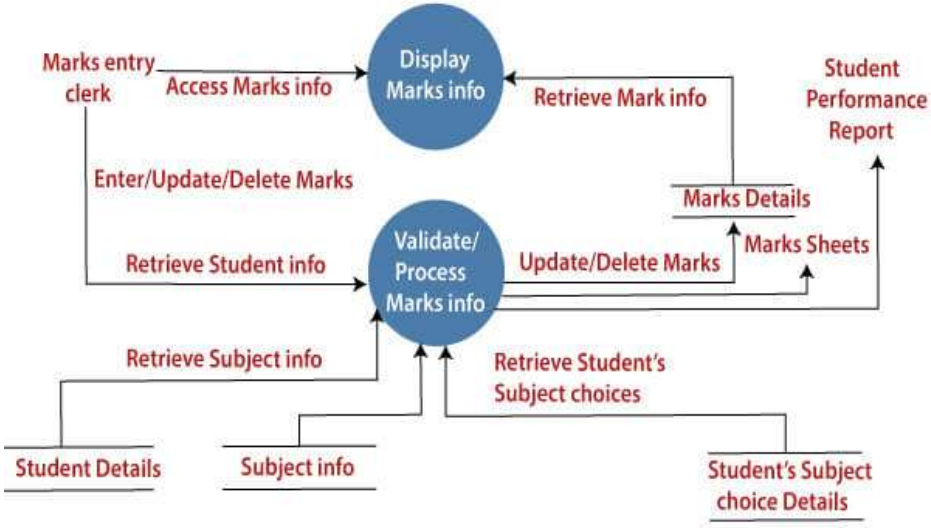
5. Student's Subject Choice Management

The Level 2 DFD of this Process is given below:



6. Marks Information Managment

The Level 2 DFD of this Process is given below:



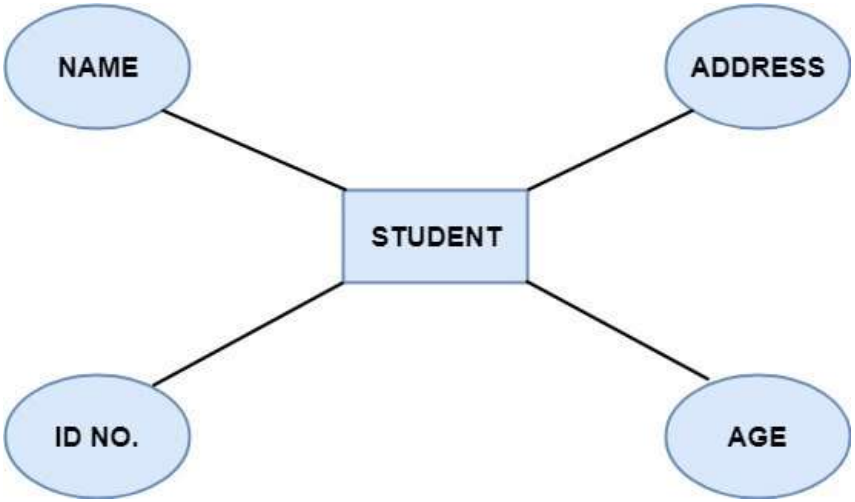
Entity Relationship Diagrams,

- ER-modeling is a data modeling method used in software engineering to produce a conceptual data model of an information system. Diagrams created using this ER-modeling method are called Entity-Relationship Diagrams or ER diagrams or ERDs.
- ERD effectively communicates the logic of the database to users.

Components of an ER Diagrams

- 1. Entity
- An entity can be a real-world object, either animate or inanimate, that can be merely identifiable. An entity is denoted as a rectangle in an ER diagram. For example, in a school database, students, teachers, classes, and courses offered can be treated as entities. All these entities have some attributes or properties that give them their identity.

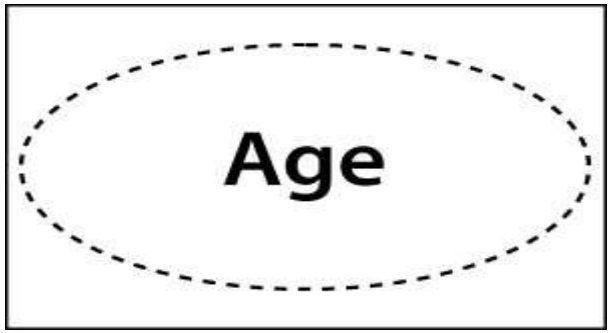
- 2. **Attributes**: Entities are denoted utilizing their properties, known as attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes.



There are four types of Attributes:

- Key attribute(Super,Candidate,Primary key)
- Composite attribute(Address[pin,ps,po])
- Single-valued attribute(Security_Number)
- Multi-valued attribute

- Derived attribute



3. Relationships

- The association among entities is known as relationship. Relationships are represented by the diamond-shaped box. For example, an employee works_at a department, a student enrolls in a course. Here, Works_at and Enrolls are called relationships.

Fig: Relationships in ERD

relationships in E-R models are:

- Unary (degree1):

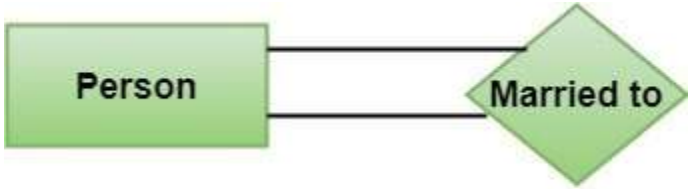


Fig: Unary Relationship

- Binary (degree2):

- Ternary (degree3):

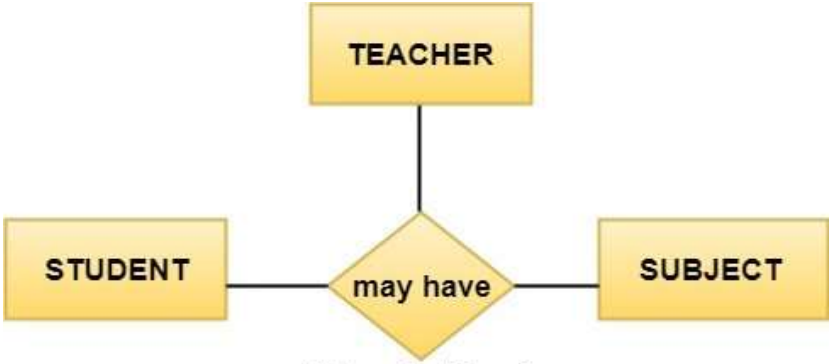


Fig: Ternary Relationship

Finalize the Requirements

- After modeling the requirements, we will have better understanding of the system behavior.
- Inconsistencies and ambiguities have been identified.
- Now we finalize the analyzed requirements and next step is to document these requirements in a prescribed format

Decision Table(CO2)

- It is a tabular form that presents a set of conditions and their corresponding actions.
- There four portion of decision table

Condition	Condition Alternatives
Actions	Actions Entries

Decision Table

- 2-Dim. Matrix:
  - Row1 : for each possible actions.
  - Row2: for each relevant conditions.
  - One column for each combination of condition states.
  - Upper rows specify the variables or conditions to be evaluated and lower rows specify the corresponding actions to be taken when an evaluation test is satisfied
  - A column in a decision table is called a rule. It implies if a condition is true then execute the corresponding action.

Decision Table

	Rules								
Conditions	**1**	**2**	**3**	**4**					n
Condition #1	✓			✓	✓				
Condition #2		✓		✓					
Condition #3			✓		✓				
Actions									
Action #1	✓			✓	✓				
Action #2		✓		✓					
Action #3			✓						
Action #4			✓	✓	✓				
Action #5	✓	✓			✓				

Decision Table for Triangle Problem

	1	2	3	4	5	6	7	8	9
C_1: X,Y,Z Triangle sides?	Y	Y	Y	Y	Y	Y	Y	Y	N
C_2: X = Y?	Y	Y	Y	Y	N	N	N	N	-
C_3: X = Z ?	Y	Y	N	N	Y	Y	N	N	-
C_4: Y = Z ?	Y	N	Y	N	Y	N	Y	N	-
A_1: not a Triangle									X
A2: Scalene								X	
A3: Isosceles				X		X	X		
A_4: Equilateral	X								
A_5: Impossible		X	X		X				

Software Quality Assurance (SQA): Verification and Validation

- The Software Quality Assurance (SQA) process compromises of the verification and validation process of the software code. In general, guaranteeing that the software conforms to its specification while meeting the customer needs. Let us compare and contrast the verification and validation process.
- Verification and validation begin by reviewing the requirements and covering the design and analysis of the code up to the product testing. For this reason, verification demands to check that the program meets specified requirements. While, on the other hand, validation requires examining that the software product meets the client expectation as well as a formal proof of program correctness

SQA Plans

- the software quality assurance plan comprises of the procedures, techniques, and tools that are employed to make sure that a product or service aligns with the requirements defined in the SRS(software requirement specification).
- **The SQA plan document consists of the below sections:**
- Purpose section
- Reference section
- Software configuration management section
- Problem reporting and corrective action section
- Tools, technologies and methodologies section
- Code control section
- Records: Collection, maintenance and retention section
- Testing methodology

software quality framework

- Software Quality Framework is a model for software quality by connecting and integrating the different views of software quality. This framework connects the developer with the customer to derive a common interpretation for quality.

ISO 9000 Models

- ISO 9000 is defined as a set of international standards on quality management and quality assurance developed to help companies effectively document the quality system elements needed to maintain an efficient quality syst

Software Engineering Institute Capability Maturity Model (SEICMM)

- The model defines a five-level evolutionary stage of increasingly organized and consistently more mature processes.

Requirement Documentation (CO2)

- The two scenario create entirely different situation and establish entirely different purposes for the document.

- First case SRS is used to define the needs and expectations of the user. Second case ,SRS is written for different purpose and serve as a contract document between customer and developer.

SRS Document(CO2)

- Requirement document is called software requirement specification(SRS). The SRS is a specification for a particular software product program or set of program that perform certain functions in a specific environment. It serves a number of purposes depending on who is writing it. First the SRS could be written by the customer of a system. Second the SRS could be written developer of a system.

Characteristics of SRS

- **Correct:** The SRS should be made up the date when appropriate requirements are identified.
- **Unambiguous:** When the requirements are correctly understood then only it is possible to write unambiguous software.
- **Complete**: To make SRS complete, its hold be specified what a software designer wants to create software.
- **Consistent:** It should be consistent with reference to the functionalities identified.
- **Specific:** The requirements should be mentioned specifically.
- **Traceable:** What is the need for mentioned requirement? This should be correctly identified.

Organization of the SRS (CO2)

- IEEE has published guidelines and standards to organize an SRS.
- First two sections are same. The specific tailoring occurs in section-3.

1. Introduction
    - 1.1 Purpose
    - 1.2 Scope
    - 1.3 Definition, Acronyms and abbreviations
    - 1.4 References
    - 1.5 Overview

Organization of the SRS

2. The Overall Description

2.1 Product Perspective

2.1.1 System Interfaces

2.1.2 Interfaces

2.1.3 Hardware Interfaces

2.1.4 Software Interfaces

2.1.5 Communication Interfaces

2.1.6 Memory Constraints

2.1.7 Operations

2.1.8 Site Adaptation Requirements

Organization of the SRS

2.2 Product Functions
2.3 User Characteristics
2.4 Constraints
2.5 Assumptions for dependencies
2.6 Apportioning of requirements

Organization of the SRS

3. Specific Requirements

3.1 External Interfaces

3.2 Functions

3.3 Performance requirements

3.4 Logical database requirements

3.5 Design Constraints

3.6 Software System attributes

3.7 Organization of specific requirements

3.8 Additional Comments.

Organization of the SRS

3. Specific Requirements

3.1 External Interfaces

3.2 Functions

3.3 Performance requirements

3.4 Logical database requirements

3.5 Design Constraints

3.6 Software System attributes

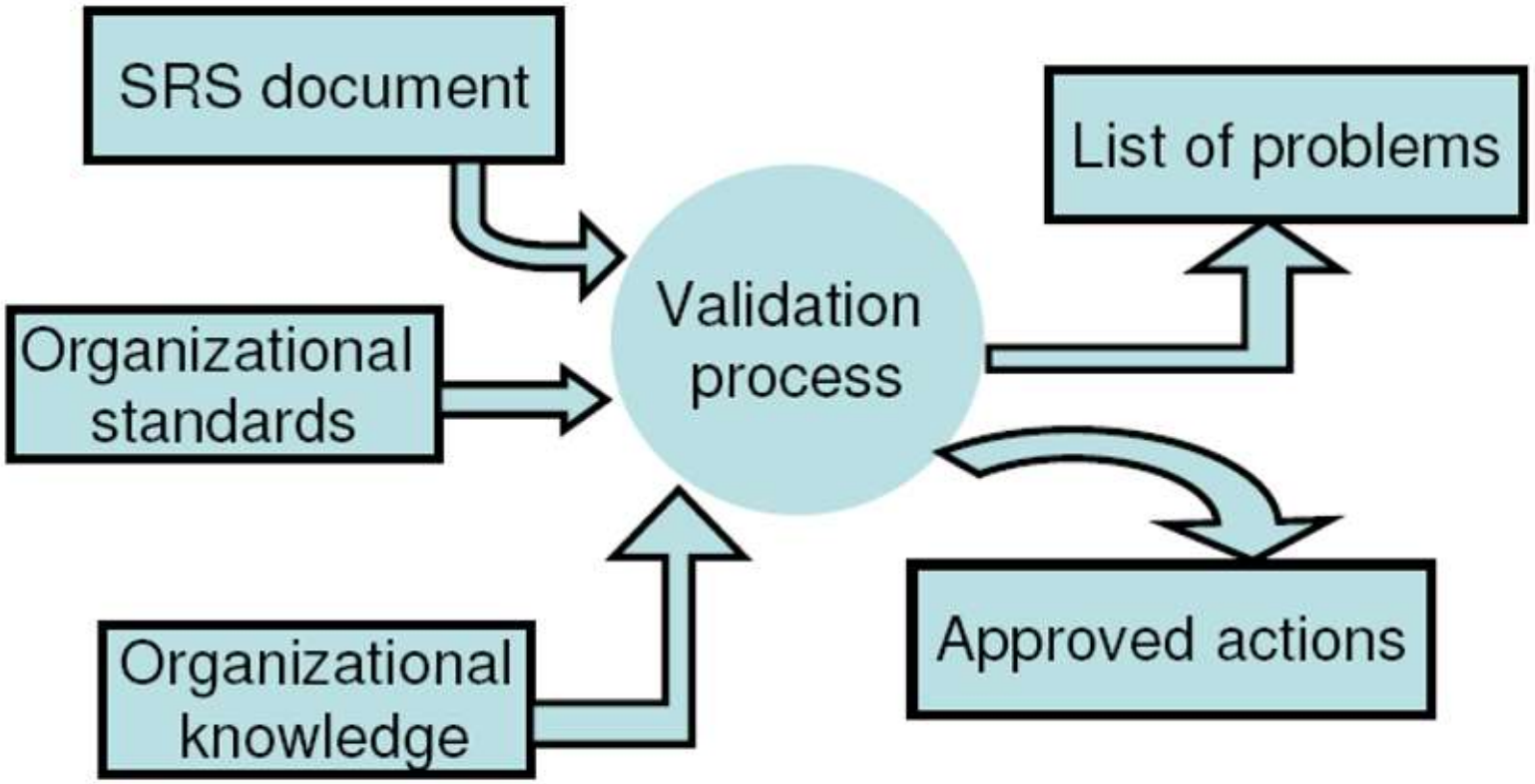
3.7 Organization of specific requirements

3.8 Additional Comments.

Requirement validation (CO2)

- After completion of SRS Check the document for
  - Completeness & consistency
  - Conformance to standards
  - Requirements conflicts
  - Technical errors
  - Ambiguous requirements
- Objective of req. validation is to certify the SRS doc., Is an acceptable doc. Of the system to be implemented.
- It find the error in doc. And improves the quality of s/w development process.
  - **Analysis**: work with row requirement as collected from various stakeholder
  - **Validation:** work with a final draft of the SRS doc. With negotiated and agreed requirements.

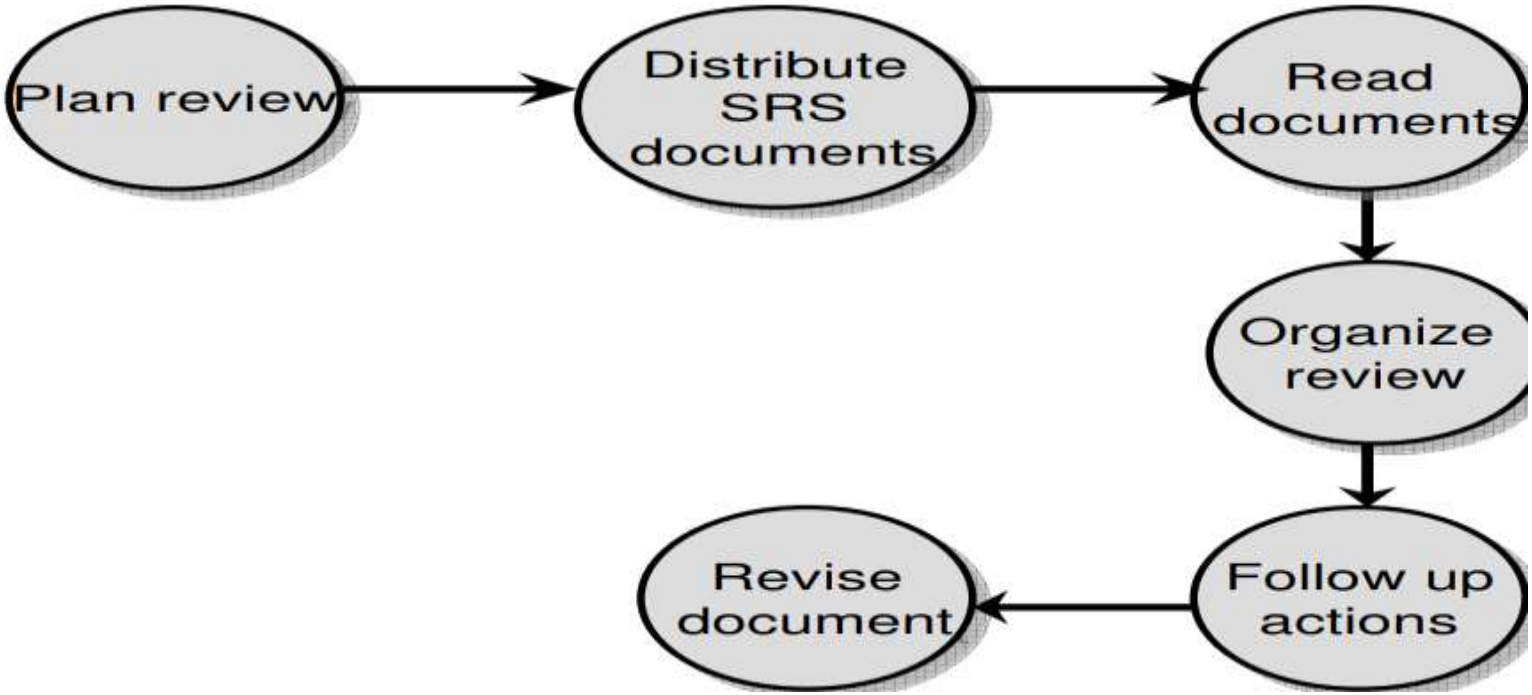
Requirement Validation



Requirement Review (CO2)

- Plan review: review team is selected, time and place is fixed for review meeting.
- Distributed SRS doc: each member should read doc. To find conflict, inconsistencies, deviations and other problems.
- Organize review meeting: each member present his/her view, problem are discussed and action on them are approved.
- Follow-up actions: chairperson of team checks that approved action have been carried out
- Revise SRS doc: SRS doc. Is revised to reflect the approved actions

Requirement Review Process



Management of User Need

- User Requirement Document(URD) :
- it is used in SE, That specify the req. the use expect from sw.
- After URD customer can not demand extra feature similarly developer cannot claim the product ready until it does not meet URD.
- Types of requirements:
  - Enduring requirements: They are core requirements & are related to main activity of the organization. Example: in library management. System issue/return of a book, cataloging etc.
  - Volatile requirements: likely to change during software development lifer cycle or after delivery of the product .

.

Management of User Need

- **Reason for change are:**
  - Change in environment.
  - Change in Technology.
  - Change in policies.
  - Change in customer's expectations.

Requirements Management

- Requirements Management : Process of understanding and controlling changes to system requirements.
- Requirement change Mgmt. process can be applied in three Stages:
  - Problem Analysis and Change Specification : change request made for partial problem then change specification are analyzed in order to validate the req. change
  - Change Analysis and Costing: estimate cost of change then take decision to implement it or not.
  - Change Implementation: when decide to change in req. Req doc. Is re-written or re-organized.

Verification and Validation(CO2)

- The purpose of V & V is to confirm system specification and to meet the requirement of system customers.
- Verification: represent the set of activities that are carried out to confirm that the s/w correctly implemented the specific functionality.(are we building the product right?)
- It is the process of determining whether the output of one phase of software development conforms to that of its previous phase. Thus verification is concerned with phase containment of errors
- Validation: represent set of activities that ensure that the s/w that has been build is satisfying the customer requirements.(are we building the right product?)
- It is the process of determining whether a fully developed system conforms to its requirements specification. the aim of validation is that the final product be error free.