

# Software Requirements Analysis and specification

# *Requirement Engineering*

---


Requirements describe

## What not How

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

Input to requirement Engineering is a problem statement prepared by the customer.

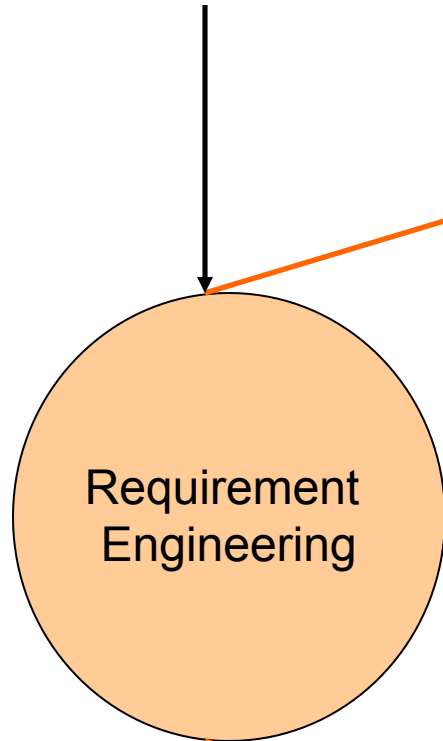
## Crucial process steps

Quality of product  Process that creates it

Without well written document

- Developers do not know what to build
- Customers do not know what to expect
- What to validate

Problem Statement



SRS

Requirements  
Elicitation

Requirements  
Analysis

Requirements  
Documentation

Requirements  
Review

Crucial Process Steps of requirement engineering

---

# *Requirement*

## *Engineering*

---

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.

### State of practice

Requirements are difficult to uncover

- Requirements change
- Over reliance on CASE Tools
- Tight project Schedule
- Communication barriers
- Market driven software development
- Lack of resources

# *Requirement*

## *Engineering*

### Example

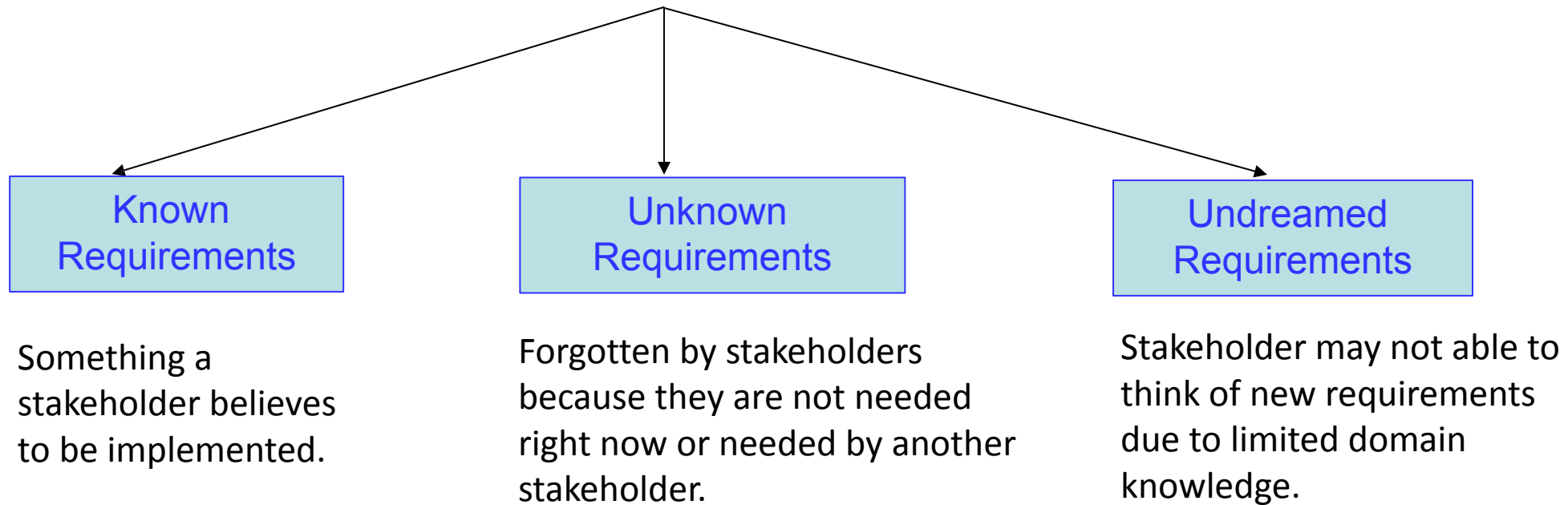
---

A University wish to develop a software system for the student result management of its M.Tech. Programme. 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.

# *Types of Requirements*

---

## Types of Requirements

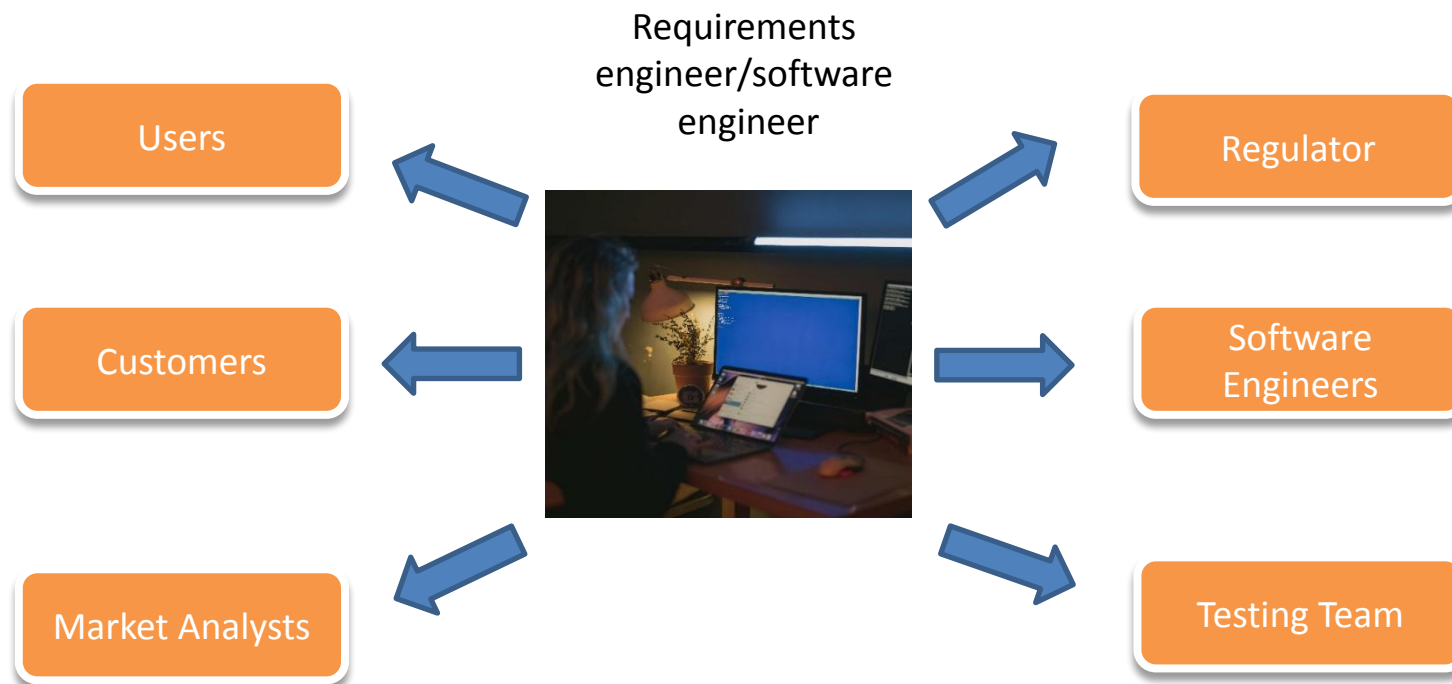


Stakeholder: Anyone who should have some direct or indirect influence on the system requirements.

--- User

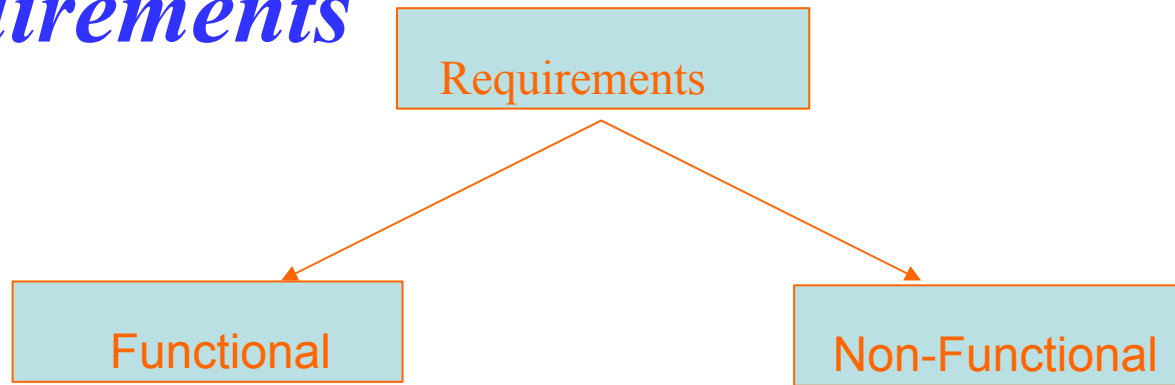
--- Affected persons

# *Process Actors / Stakeholders*



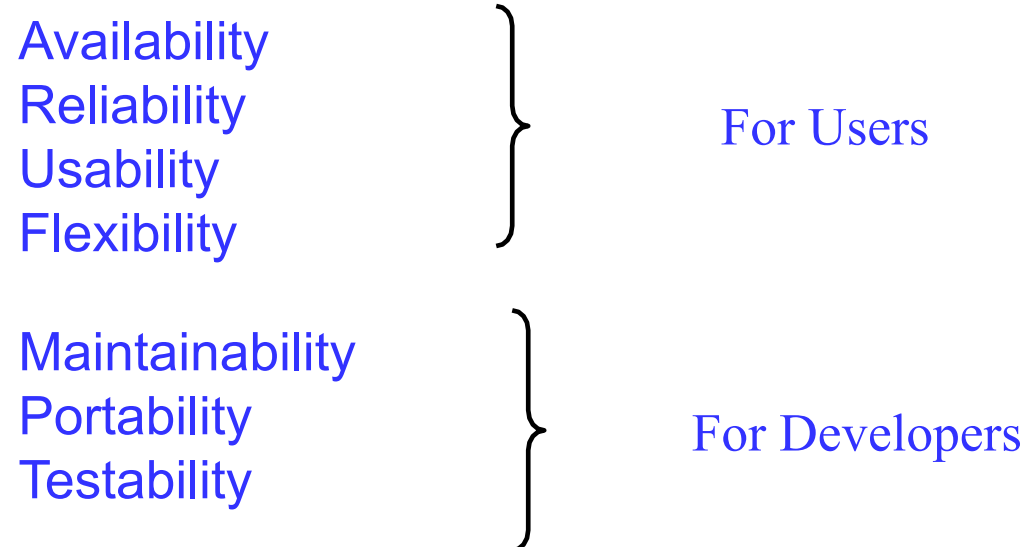
# *Types of Requirements*

---



**Functional requirements** describe what the software has to do. They are often called product features.

**Non Functional requirements** are mostly quality requirements. That stipulates how well the software does, and what it has to do.





# *Functional vs. Non-Functional Requirements*

## Functional requirements

- Describe the functions that the software is to execute
- Also known as "capabilities" or "features"

## Non-Functional Requirements (NFRs)

- Non-functional requirements are the ones that act to constrain the solution .
- Also known as "constraints" or "quality requirements"

# *Exercise: Understanding NFRs*

## **Non-Functional Requirement Name**

A. Reusability

B. Efficiency

C. Testability

D. Portability

E. Maintainability

F. Usability

G. Robustness

## **Description**

1. How well the system uses the processor capacity  
space, communication, or band width

2. How easy it is to correct a defect or make a change to the software

3. Extent to which a component designed for one application can be used in an totally different application

4. The ease with which the software components or integrated products can be tested

5. Effort required to move or migrate software from one OS to an other

6. Degree to which the system continues to function correctly when confronted with invalid data

7. Ease to use and human engineering, user friendliness

# *Exercise: Understanding NFRs*

<b>Non-Functional Requirement Name</b>	<b>Description</b>
A. Reusability	3. Extent to which a component designed for one application can be used in an totally different application
B. Efficiency	1. How well the system uses the processor capacity, disk space, communication, or band width
C. Testability	4. The ease with which the software components or integrated products can be tested
D. Portability	5. Effort required to move or migrate software from one OS to an other
E. Maintainability	2. How easy it is to correct a defect or make a change to the software
F. Usability	7. Ease to use and human engineering, user friendliness
G. Robustness	6. Degree to which the system continues to function correctly when confronted with invalid data

# *Functional Requirements: Examples*

In airline tracking system, the system shall assign a unique PNR number to each booking.

FastTag in electronic Toll collection system assign UPI Id for making toll payments from customers linked prepaid or saving account.

# *Non Functional Requirements: Examples*

In Airline Tracking System

“Given a PNR number as input, the system should display the status of the flight within 3 seconds to the user (performance requirement).

“The System shall protect against unauthorised addition or deletion of PNR number.” (Security Requirement)

# *Product vs. Process Requirements*

## Product requirements

- What the software must do or what users must be able to do with it

## Process requirements

- Constraints on development of the software (including specification of a development language or toolset, verification techniques, or the overall process to be followed)

# *Product Requirements: Examples*

For a course registration system in a university: "The software should verify that a student meets all prerequisites before he or she registers for a course"

For a telecommunications billing application:

- "The software should be able to generate 100 bills per second (performance requirement)"
- "The software must support Mac OS X and Windows 8.1 and later versions (portability requirement)"

# *Process Requirements: Examples*

For a software embedded in a router device: “The software must be implemented in C language (example of a language constrain)”

For a Networking Management System (NMS) software:  
“The software must be implemented using Rational Unified Process (RUP) method (example of a process constraint)”



# *Types of*

## *Requirements*

---

### User and system requirements

- User requirement are written for the users and include functional and non functional requirement.
- System requirement are derived from user requirement.
- The user system requirements are the parts of software requirement and specification (SRS) document.

# *Types of*

## *Requirements* Interface Specification

---

- Important for the customers.

### TYPES OF INTERFACES

- Procedural interfaces (also called Application Programming Interfaces (APIs)).
- Data structures
- Representation of data.

# *Feasibility*

---

## *Study*

### **Is cancellation of a project a bad news?**

As per IBM report, “31% projects get cancelled before they are completed, 53% over-run their cost estimates by an average of 189% & for every 100 projects, there are 94 restarts.

### **How do we cancel a project with the least work?**



**CONDUCT A FEASIBILITY STUDY**

# *Feasibility*

---

## *Study*

### Technical feasibility

- Is it technically feasible to provide direct communication connectivity through space from one location of globe to another location?
- Is it technically feasible to design a programming language using “Sanskrit”?

# *Feasibility*

---

## *Study*

Feasibility depends upon non technical Issues like:

- Are the project's cost and schedule assumption realistic?
- Does the business model realistic?
- Is there any market for the product?

# *Feasibility*

---

## *Study*

### Purpose of feasibility study

“evaluation or analysis of the potential impact of a proposed project or program.”

### Focus of feasibility studies

- Is the product concept viable?
- Will it be possible to develop a product that matches the project's vision statement?
- What are the current estimated cost and schedule for the project?

# *Feasibility*

## *Study*

### Focus of feasibility studies

- How big is the gap between the original cost & schedule targets & current estimates?
- Is the business model for software justified when the current cost & schedule estimate are considered?
- Have the major risks to the project been identified & can they be surmounted?
- Is the specifications complete & stable enough to support the remaining development work?
- Have users & developers been able to agree on a detailed user interface prototype? If not, are the requirements really stable?
- Is the software development plan complete & adequate to support further development work?

# *Requirements*

## *Elicitation*

---

Perhaps

- Most difficult
- Most critical
- Most error-prone
- Most communication intensive

Succeed



effective customer developer partnership

Selection of any method

1. It is the only method that we know
2. It is our favorite method for all situations
3. We understand intuitively that the method is effective in the present circumstances.

Normally we rely on the first two reasons.



# *Methods for Requirements Elicitation*

---

## **1. Interviews**     --- open ended                                  --- structured

- After receiving the problem statement from the customer the first step is to arrange a meeting with the customer.
- The objective of the interview is to understand the customer's expectations of the software.
- In an **Open-ended interview**, there is no present agenda, and context-free questions may be asked to understand the problem and overview the situation.
- In the **case of structured interviews** the agenda of questions are prepared or a proper set of questions are designed for the interview.
- Selection of stakeholder
  1. Entry level personnel
  2. Middle level stakeholder
  3. Managers
  4. Users of the software (Most important)

# *Requirements*

---

## *Elicitation*

Types of questions:

- Any problems with the existing system
- Any Calculation errors
- Possible reasons for malfunctioning
- No. of Student Enrolled
- Possible benefits
- Satisfied with current policies
- How are you maintaining the records of previous students?
- Any requirement of data from other systems
- Any specific problems
- Any additional functionality
- Most important goal of the proposed development

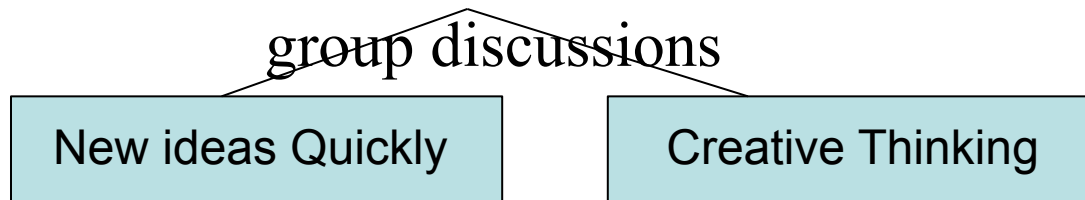
At the end, we may have a wide variety of expectations from the proposed software.

# *Requirements*

---

## *Elicitation* 2. Brainstorming Sessions

It is a group technique that may be used for requirement elicitation to understand the requirements. The group Discussion may leads to new ideas and help to promote creative thinking.



Prepare long list of requirements

└─┬─> Categorized  
    Prioritized  
    Pruned

\*Idea is to generate views , not to vet them.

### Groups

1. Users 2. Middle Level managers 3. Total Stakeholders

# Requirements

## Elicitation

A Facilitator may handle group bias, and conflicts carefully.

- Facilitator may follow a published agenda
- Every idea will be documented in a way that everyone can see it.
- A detailed report is prepared.

### 3. Facilitated Application specification Techniques (FAST)

- Similar to brainstorming sessions.
- Team-oriented approach
- Creation of a joint team of customers and developers.
- The objective is to bridge the expectation gap which is the difference between what stakeholders think they are supposed to build and what customers think they are going to get.

# *Requirements*

---

## *Elicitation* Guidelines

1. Arrange a meeting at a neutral site.
2. Establish rules for participation.
3. Informal agenda to encourage free flow of ideas.
4. Appoint a facilitator.
5. Prepare definition mechanism board, worksheets, wall stickier.
6. Participants should not criticize or debate.

## FAST session Preparations

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

# *Requirements*

## *Elicitation*

---

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

### Activities of FAST session

1. Every participant presents his/her list
2. Combine list for each topic
3. Discussion
4. Consensus list
5. Sub teams for mini specifications
6. Presentations of mini-specifications
7. Validation criteria
8. A sub team to draft specifications

# Requirements

---

## Elicitation

### 4. Quality Function Deployment

- Incorporate voice of the customer



Technical requirements.



Documented

Prime concern is customer satisfaction



What is important for customer?

- Normal requirements
- Expected requirements
- Exciting requirements

# *Requirements*

---

## *Elicitation*

### Steps

1. Identify stakeholders
2. List out requirements
3. Degree of importance to each requirement.



# Requirements

---

## Elicitation

- |          |   |   |
|----------|---|---|
| 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 |

Requirement Engineer may 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.

# Requirements

## Elicitation

### 5. The Use Case Approach

Ivar Jacobson & others introduced Use Case approach for elicitation & modeling.

Use Case – give functional view

The terms

Use Case

Use Case Scenario

Use Case Diagram

} Often Interchanged

} But they are different

Use Cases are structured outline or template for the description of user requirements modeled in a structured language like English.

Use case Scenarios are unstructured descriptions of user requirements.

Use case diagrams are graphical representations that may be decomposed into further levels of abstraction.

# Requirements

## Elicitation

---

### Components of Use Case approach

Actor: An actor or external agent, lies outside the system model, but interacts with it in some way.

Actor → Person, machine, information System

- Cockburn distinguishes between Primary and secondary actors.
- A Primary actor (Active actors) is one having a goal requiring the assistance of the system.
- A Secondary actor (Passive Actors) is one from which the System needs assistance.

### Use Cases

A use case is initiated by a user with a particular goal in mind, and completes successfully when that goal is satisfied.

# *Requirements*

## *Elicitation*

---

- \* It describes the sequence of interactions between actors and the system necessary to deliver the services that satisfies the goal.
- \* Alternate sequence
- \* System is treated as black box.

Thus

Use Case captures who (actor) does what (interaction) with the system, for what purpose (goal), without dealing with system internals.

# Requirements

## ~~Elicitation~~

\*defines all behavior required of the system, bounding the scope of the system.

Jacobson & others proposed a template for writing Use

cases as shown below:

### 1. Introduction

Describe a quick background of the use case.

### 2. Actors

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

### 3. Pre Conditions

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

### 4. Post Conditions

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

# Requirements

---

## *Elicitation*

### 5. Flow of events

#### 1. Basic Flow

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

#### 2. 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.

### 6. Special Requirements

Business rules should be listed for basic & information flows as special requirements in the use case narration .These rules will also be used for writing test cases. Both success and failures scenarios should be described.

### 7. Use Case relationships

For Complex systems it is recommended to document the relationships between use cases. Listing the relationships between use cases also provides a mechanism for traceability

# *Requirements*

## *Elicitation*

---

### Use Case Guidelines

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 same level of abstraction throughout the use case. Steps in higher level use cases may be treated as goals for lower level (i.e. more detailed), sub-use cases.
4. Structure the use case
5. Review and validate with users.

# Requirements

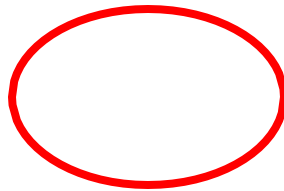
## ~~Elicitation~~

### Use case Diagrams

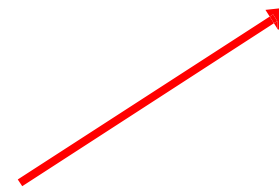
- represents what happens when actor interacts with a system.
- captures functional aspect of the system.



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.



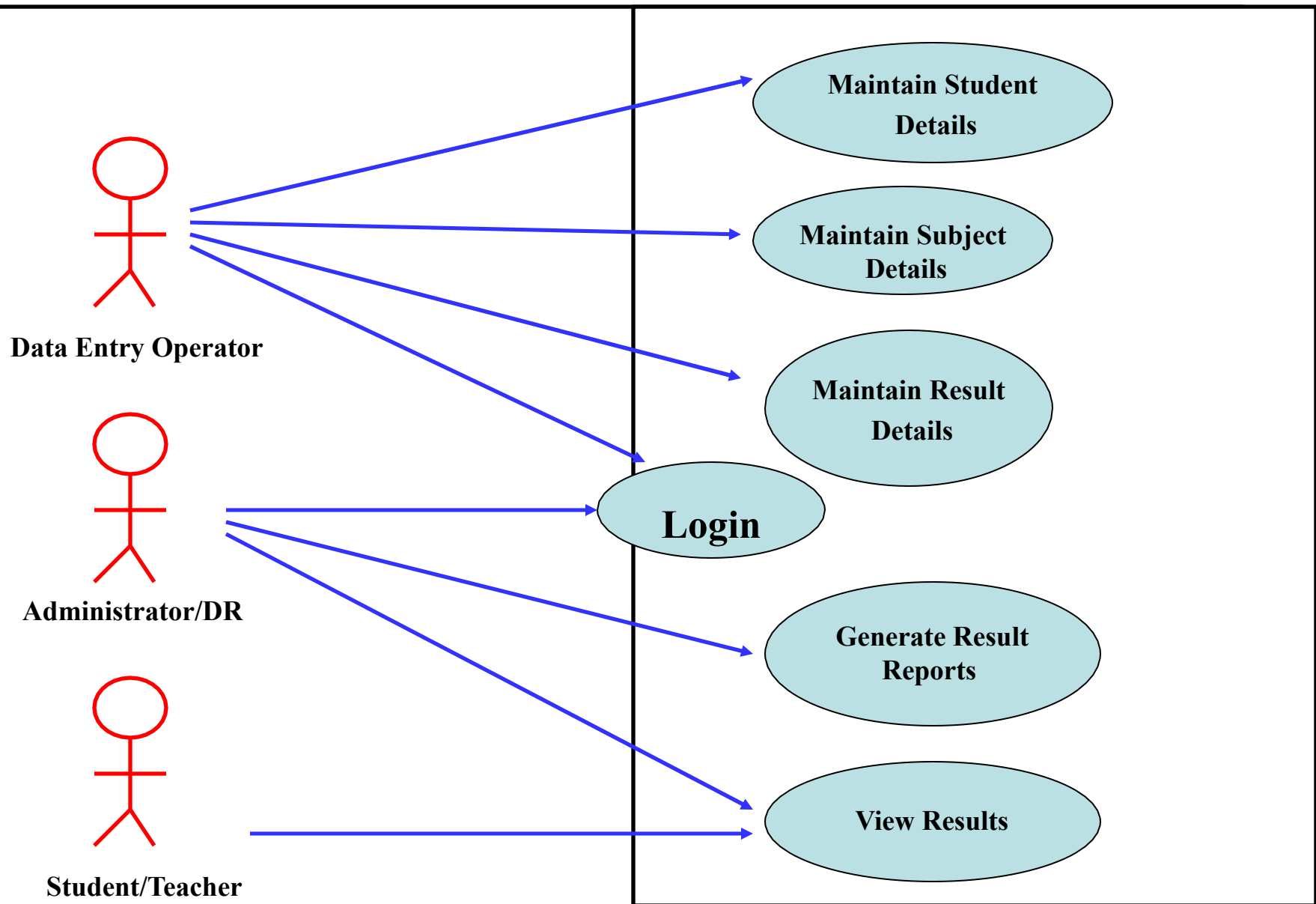
# *Requirements*

## *Elicitation*

---

- \* Use cases should not be used to capture all the details of the system.
- \* Only significant aspects of the required functionality
- \* No design issues
- \* Use Cases are for “what” the system is , not “how” the system will be designed
- \* Free of design characteristics

# Use case diagram for Result Management System



# Requirements

---

## 1. *Elicitation* Maintain Student Details

Add/Modify/update students details like name, address.

## 2. Maintain subject Details

Add/Modify/Update Subject information semester wise

## 3. Maintain Result Details

Include entry of marks and assignment of credit points for each paper.

## 4. Login

Use to Provide way to enter through user id & password.

## 5. Generate Result Report

Use to print various reports

## 6. View Result

- (i) According to course code
- (ii) According to Enrollment number/roll number

# *Requirements Elicitation (Use Case)*

---

## 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

3. Pre Conditions : None

4. Post Conditions : If the use case is successful, the actor is logged into the system. If not, the system state is unchanged.

# *Requirements Elicitation (Use Case)*

---

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.

# Use

## Cases

### 6. Alternate Flows

#### 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.

#### 7. Special Requirements:

None

#### 8. Use case Relationships:

None

# Use

## Cases

---

### 2. Maintain student details

2.1 Introduction : Allow DEO to maintain student details.  
This includes adding, changing and deleting student information

2.2 Actors : DEO

2.3 Pre-Conditions: DEO must be logged onto the system before this use case begins.

# Use

## Cases

---

2.4 Post-conditions : If use case is successful, information is added/updated/deleted from the system. Otherwise, the system state is unchanged.

5. Basic Flow : Starts when DEO wishes to add/modify/update/delete Student information.

(i) The system requests the DEO to specify the function, he/she would like to perform (Add/update/delete)

(ii) One of the sub flow will execute after getting the information.



# Use

## Cases

---

- ❑ If DEO selects "Add a student", "Add a student" sub flow will be executed.
- ❑ If DEO selects "update a student", "update a student" sub flow will be executed.
- ❑ If DEO selects "delete a student", "delete a student" sub flow will be executed.

### 1. Add a student

- (i) The system requests the DEO to enter:  
Name  
Address  
Roll No  
Phone No  
Date of admission

- (ii) System generates unique id

# Use

## Cases

---

### 2.5.2 Update a student

- (i) System requires the DEO to enter student-id.
- (ii) DEO enters the student\_id. The system retrieves and display the student information.
- (iii) DEO make the desired changes to the student information.
- (iv) After changes, the system updates the student record with changed information.

### 2.5.3 Delete a student

- (i) The system requests the DEO to specify the student-id.
- (ii) DEO enters the student-id. The system retrieves and displays the student information.
- (iii) The system prompts the DEO to confirm the deletion of the student.
- (iv) The DEO confirms the deletion.
- (v) The system marks the student record for deletion.

# Use

## Cases

---

### 6. Alternative flows

#### 1. Student not found

If in the update a student or delete a student sub flows, a student with specified\_id does not exist, the system displays an error message .The DEO may enter a different id or cancel the operation. At this point ,Use case ends.

*Use*

*Cases*

---

### 2.6.2 Update Cancelled

If in the update a student sub-flow, the data entry operator decides not to update the student information, the update is cancelled and the basic flow is restarted at the begin.

# *Use*

## *Cases*

---

### 2.6.3 Delete cancelled

If in the delete a student sub flows, DEO decides not to delete student record ,the delete is cancelled and the basic flow is re-started at the beginning.

#### 7. Special requirements

None

#### 8. Use case relationships

None

### 3. Maintain Subject Details

#### 3.1 Introduction

The DEO to maintain subject information. This includes adding, changing, deleting subject information from the system

2. Actors : DEO

3. Preconditions : DEO must be logged onto the system before the use case begins.

# Use

## Cases

---

### 4. Post conditions :

If the use case is successful, the subject information is added, updated, or deleted from the system, otherwise the system state is unchanged.

### 5. Basic flows :

The use case starts when DEO wishes to add, change, and/or delete subject information from the system.

(ii) The system requests DEO to specify the function he/she would like to perform i.e.

- Add a subject
- Update a subject
- Delete a subject.



# Use

## Cases

(ii) Once the DEO provides the required information, one of the sub flows is executed.

- ❑ If DEO selected “Add a subject” the “Add-a subject sub flow is executed.
- ❑ If DEO selected “Update-a subject” the “update-a- subject” sub flow is executed
- ❑ If DEO selected “Delete- a- subject”, the “Delete-a-subject” sub flow is executed.

### 3.5.1 Add a Subject

- (i) The System requests the DEO to enter the subject information. This includes :
  - \* Name of the subject

# *Use*

---

## *Cases*

- \* Subject Code
- \* Semester
- \* Credit points

- (ii) Once DEO provides the requested information, the system generates and assigns a unique subject-id to the subject. The subject is added to the system.
- (iii) The system provides the DEO with new subject-id.

### 3.5.2 Update a Subject

- (i) The system requests the DEO to enter subject\_id.
- (ii) DEO enters the subject\_id. The system retrieves and displays the subject information.
- (iii) DEO makes the changes.
- (iv) Record is updated.

# *Use*

## *Cases*

---

### 3. Delete a Subject

- (i) Entry of subject\_id.
- (ii) After this, system retrieves & displays subject information.
  - \* System prompts the DEO to confirm the deletion.
  - \* DEO verifies the deletion.
  - \* The system marks the subject record for deletion.

### 6. Alternative Flow

#### 1. Subject not found

If in any sub flows, subject-id not found, error message is displayed. The DEO may enter a different id or cancel the case ends here.

#### 2. Update Cancelled

If in the update a subject sub-flow, the data entry operator decides not to update the subject information, the update is cancelled and the basic flow is restarted at the begin.

# Use Cases

---

## 3.6.3 Delete Cancellation

If in delete-a-subject sub flow, the DEO decides not to delete subject, the delete is cancelled, and the basic flow is restarted from the beginning.

3.7 Special

Requirements:

None

3.8 Use Case-relationships

None

### 4. Maintain Result Details

#### 1. Introduction

This use case allows the DEO to maintain subject & marks information of each student. This includes adding and/or deleting subject and marks information from the system.

#### 4.2 Actor

DEO

# *Use*

## *Cases*

---

### 4.3 Pre Conditions

DEO must be logged onto the system.

### 4.4 Post Conditions

If use case is successful ,marks information is added or deleted from the system. Otherwise, the system state is unchanged.



### 5. Basic Flow

This use case starts, when the DEO wishes to add, update and/or delete marks from the system.

- (i) DEO to specify the function
- (ii) Once DEO provides the information one of the subflow is executed.
  - \* If DEO selected “Add Marks”, the Add marks subflow is executed.
  - \* If DEO selected “Update Marks”, the update marks subflow is executed.
  - \* If DEO selected “Delete Marks”, the delete marks subflow is executed.

# *Use*

---

## *Cases*

### 1. Add Marks Records

Add marks information .This includes:

- a. Selecting a subject code.
- b. Selecting the student enrollment number.
- c.Entering internal/external marks for that subject code & enrollment number.

# Use

## Cases

---

- (ii) If DEO tries to enter marks for the same combination of subject and enrollment number, the system gives a message that the marks have already been entered.
- (iii) Each record is assigned a unique result\_id.

## 2. Delete Marks records

### 1. DEO makes the following entries:

- a. Selecting subject for which marks have to be deleted.
- b. Selecting student enrollment number.
- c. Displays the record with id number.
- d. Verify the deletion.
- e. Delete the record.

### 2. Update Marks records

- 1.The System requests DEO to enter the record\_id.
- 2.DEO enters record\_id. The system retrieves & displays the information.
3. DEO makes changes.
4. Record is updated.

# Use

---

## <sup>3</sup>Cases Compute Result

- (i) Once the marks are entered, result is computed for each student.
- (ii) If a student has scored more than 50% in a subject, the associated credit points are allotted to that student.
- (iii) The result is displayed with subject-code, marks & credit points.

## 7. Alternative Flow

### 1. Record not found

If in update or delete marks sub flows, marks with specified id number do not exist, the system displays an error message. DEO can enter another id or cancel the operation.

*Use*

*Cases*

---

## 4.6.2 Delete Cancelled

If in Delete Marks, DEO decides not to delete marks, the delete is cancelled and basic flow is re-started at the beginning.

## 7. Special Requirements

None

## 8. Use case relationships

None

# Use

## Cases

---

### 5 View/Display result

#### 1. Introduction

This use case allows the student/Teacher or anyone to view the result. The result can be viewed on the basis of course code and/or enrollment number.

#### 2. Actors

Administrator/DR, Teacher/Student

#### 3. Pre

Conditions

None

#### 4. Post Conditions

If use case is successful, the marks information is displayed by the system. Otherwise, state is unchanged<sup>71</sup>

# *Use*

## *Cases*

---

### 5.5 Basic Flow

Use case begins when student, teacher or any other person wish to view the result.

Two ways

- Enrollment no.

- Course code



# Use

## Cases

---

(ii) After selection, one of the sub flow is executed.

Course code            →            Sub flow is executed

Enrollment no.        →            Sub flow is executed

1.        View result enrollment number wise

(i)    User to enter enrollment number

(ii)   System retrieves the marks of all subjects with credit points

(iii)   Result is displayed.

# *Use*

---

## *Cases*

### 6. Alternative Flow

1. Record not found  
Error message should be displayed.

### 7. Special Requirements

None

### 8. Use Case relationships

None

# Use

## Cases Generate Report

---

### 1. Introduction

This use case allows the DR to generate result reports. Options are

- a. Course code wise
- b. Semester wise
- c. Enrollment Number wise

### 2. Actors

DR

### 3. Pre-Conditions

DR must logged on to the system

# Use

## Cases

---

### 4. Post conditions

If use case is successful, desired report is generated. Otherwise, the system state is unchanged.

### 5. Basic Flow

The use case starts, when DR wish to generate reports.

- (i) DR selects option.
- (ii) System retrieves the information displays.
- (iii) DR takes printed reports.

# Use

## Cases

---

### 6. Alternative Flows

#### 1. Record not found

If not found, system generates appropriate message. The DR can select another option or cancel the operation. At this point, the use case ends.

### 6.7 Special Requirements

None

### 6.8 Use case relationships

None

## Cases 7. Maintain User Accounts

---

### 1. Introduction

This use case allows the administrator to maintain user account. This includes adding, changing and deleting user account information from the system.

### 2. Actors

Administrator

### 3. Pre-Conditions

The administrator must be logged on to the system before the use case begins.

# Use

## Cases

---

### 4. Post-Conditions

If the use case was successful, the user account information is added, updated, or deleted from the system. Otherwise, the system state is unchanged.

### 5. Basic Flow

This use case starts when the Administrator wishes to add, change, and/or delete use account information from the system.

- (ii) The system requests that the Administrator specify the function he/she would like to perform (either Add a User Account, Update a User Account, or Delete a User Account).
- (iii) Once the Administrator provides the requested information, one of the sub-flows is executed

# Use

## Cases

- \* If the Administrator selected “Add a User Account”, the **Add a User Account** sub flow is executed.
- \* If the Administrator selected “Update a User Account”, the **Update a User Account** sub-flow is executed.
- \* If the Administrator selected “Delete a User Account”, the **Delete a User Account** sub-flow is executed.22

### 7.5.1 Add a User Account

1.The system requests that the Administrator enters the user information. This includes:

- (a) User Name
- (b) User ID-should be unique for each user account
- (c) Password
- (d) Role



# Use

## Cases

---

2. Once the Administrator provides the requested information, the user account information is added.

### 7.5.2 Update a User Account

1. The system requests that the Administrator enters the User ID.
2. The Administrator enters the User ID. The system retrieves and displays the user account information.
3. The Administrator makes the desired changes to the user account information. This includes any of the information specified in the **Add a User Account** sub-flow.
4. Once the Administrator updates the necessary information, the system updates the user account record with the updated information.

### 3. Delete a User Account

1. The system requests that the Administrator enters the User ID.
2. The Administrator enters the User ID. The system retrieves and displays the user account information.
3. The system prompts the Administrator to confirm the deletion of the user account.
4. The Administrator confirms the deletion.
5. The system deletes the user account record.

### 7.6 Alternative Flows

#### 7.6.1 User Not Found

If in the **Update a User Account** sub-flow, a specified **User ID** does not exist, the system displays an error message. The Administrator can then enter a different User ID or cancel the operation, at which point the use case ends.

#### 7.6.2 Update Cancelled

If in the **Update a User Account** sub-flow, the Administrator decides not to update the user account information, the update is cancelled and the **Basic Flow** is re-started at the beginning.

### 7.6.3 Delete Cancelled

If in the **Delete a User Account** sub-flow, the Administrator decides not to delete the user account information, the delete is cancelled and the **Basic Flow** is re-started at the beginning.

### 7.7 Special Requirements

None

### 7.8 Use case relationships

None

## Cases

---

### 8. Reset System

#### 1. Introduction

This use case allows the administrator to reset the system by deleting all existing information from the system .

#### 2. Actors

Administrator

#### 3. Pre-Conditions

The administrator must be logged on to the system before the use case begins.

### 4. Post-Conditions

If the use case was successful, all the existing information is deleted from the backend database of the system. Otherwise, the system state is unchanged.

### 5. Basic Flow

This use case starts when the Administrator wishes to reset the system.

- i. The system requests the Administrator to confirm if he/she wants to delete all the existing information from the system.
- ii. Once the Administrator provides confirmation, the system deletes all the existing information from the backend database and displays an appropriate message.

# Use

## Cases

---

### 6. Alternative Flows

#### 1. Reset Cancelled

If in the Basic Flow, the Administrator decides not to delete the entire existing information, the reset is cancelled and the use case ends.

### 7. Special Requirements

None

### 8.8 Use case relationships

None

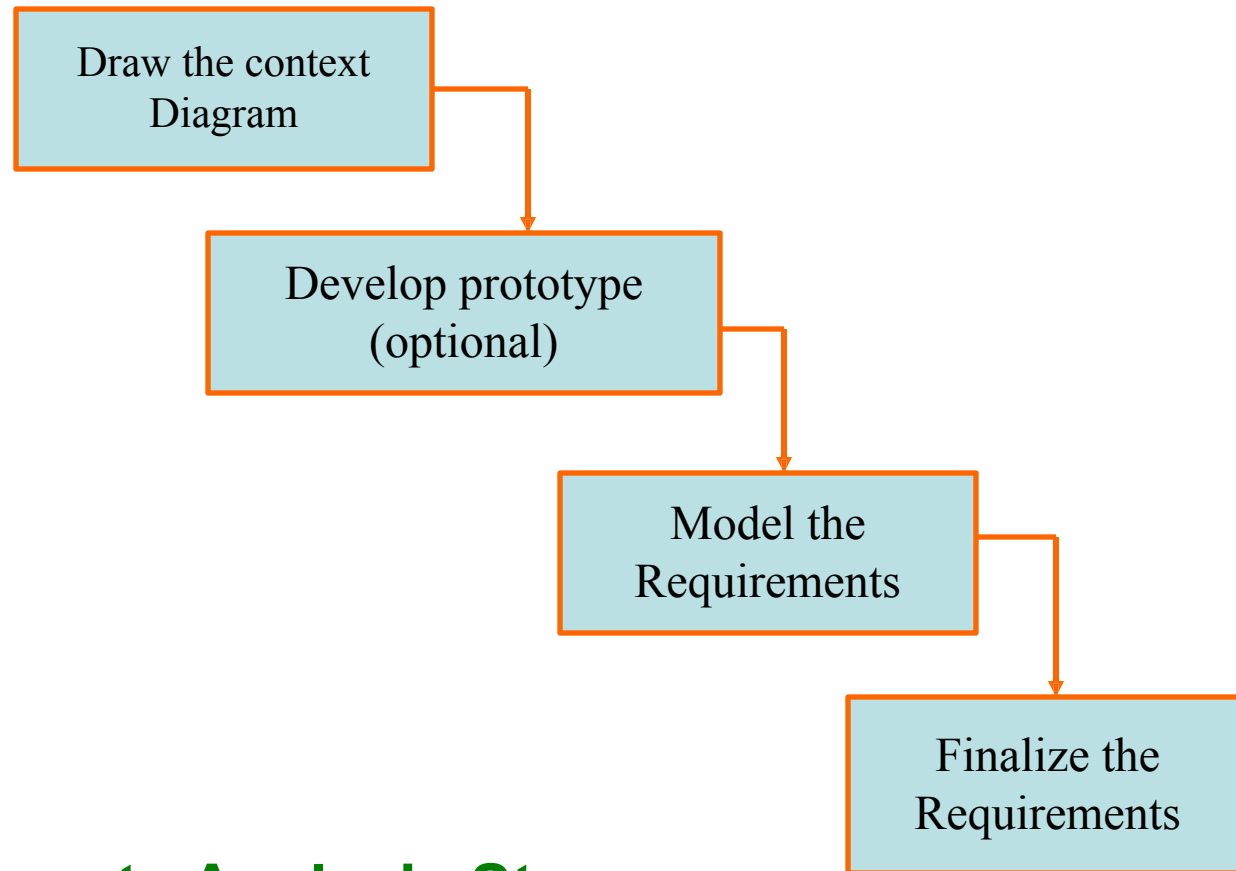
# Requirements

---

## Analysis

We analyze, refine and scrutinize requirements to make consistent & unambiguous requirements.

### Steps

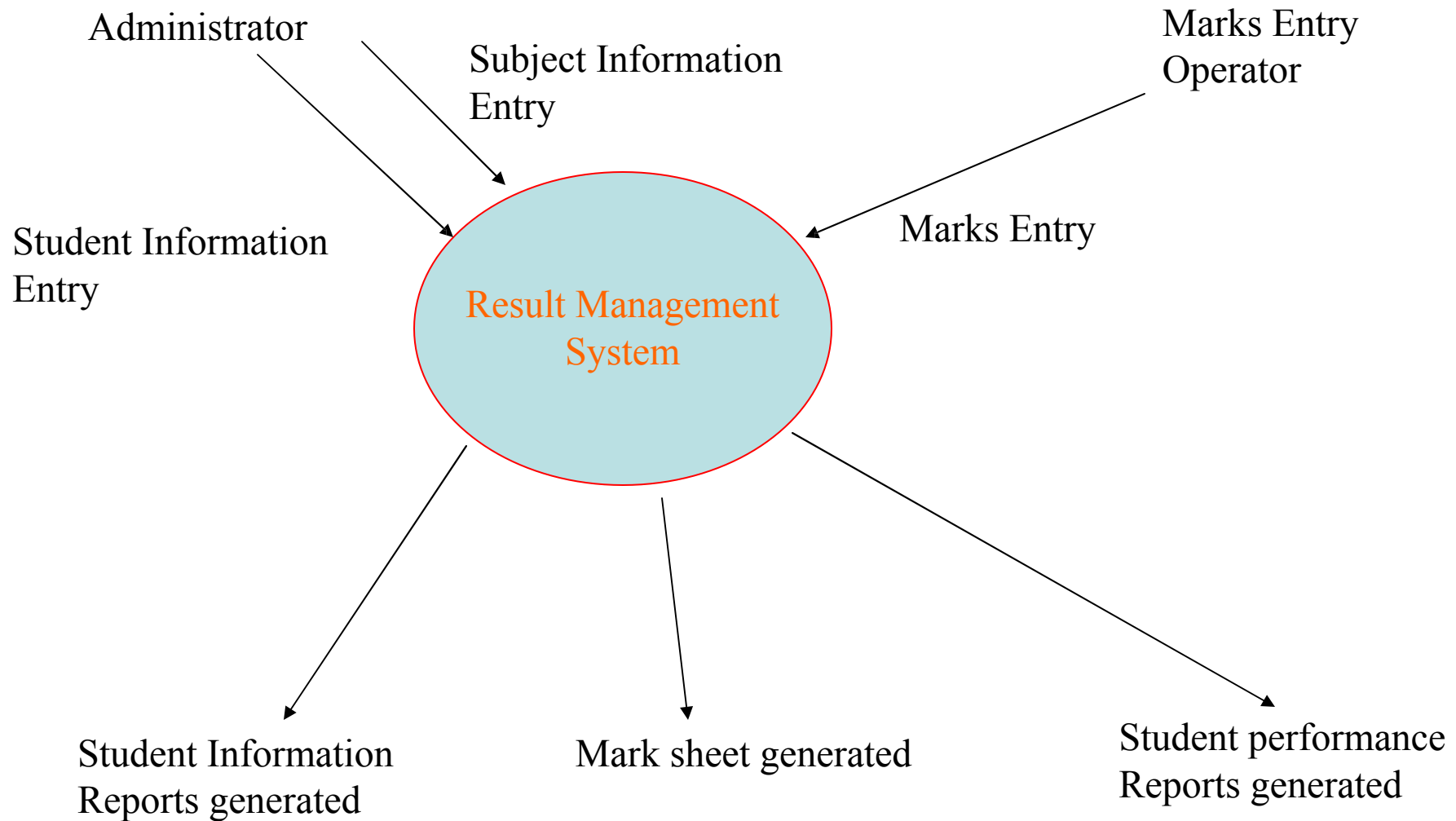


### Requirements Analysis Steps



# Requirements Analysis

---



# Requirements

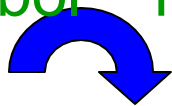
## Analysis

### Data Flow Diagrams

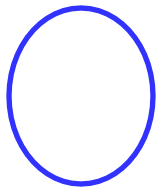
DFD show the flow of data through the system.

- All names should be unique
- It is not a flow chart
- Suppress logical decisions
- Defer error conditions & handling until the end of the analysis

Symbol      Name      Function



Data Flow      Connect process


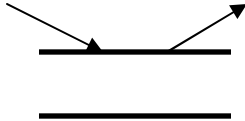


Process

Perform some transformation of its input data to yield output data.

# Requirements

## Analysis

Symbol	Name	Function
	Source or sink	A source of system inputs or sink of system outputs
	Data Store	A repository of data, the arrowhead indicate net input and net outputs to store

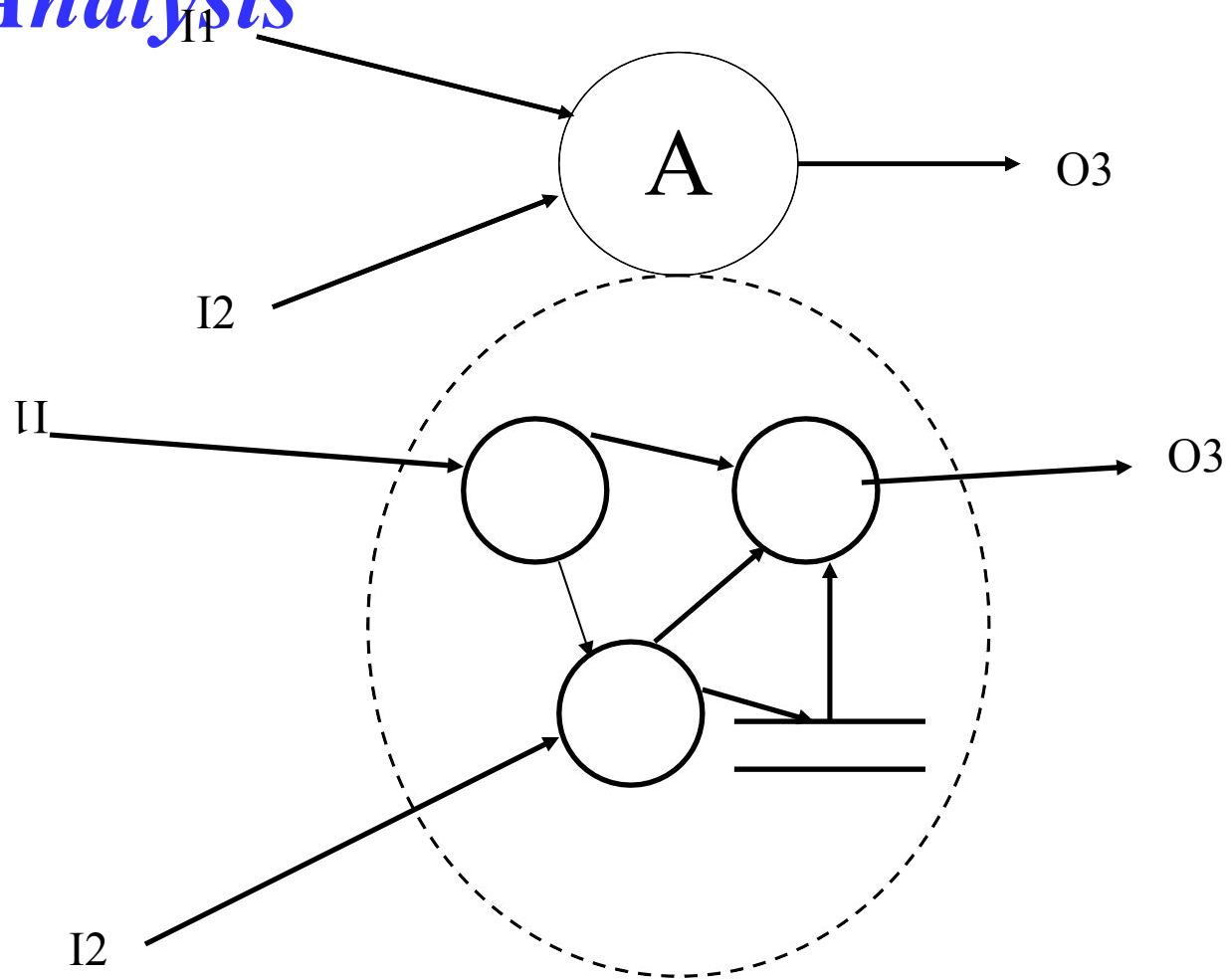
## Leveling

DFD represent a system or software at any level of abstraction.

A level 0 DFD is called fundamental system model or context model represents entire software element as a single bubble with input and output data indicating by incoming & outgoing arrows.

# Requirements

## Analysis



# *Data*

## *Dictionaries*

---

DFD   DD →

Data Dictionaries are simply repositories to store information about all data items defined in DFD.

Includes :

- Name of data item
- Aliases (other names for items)
- Description/Purpose
- Related data items
- Range of values
- Data flows
- Data structure definition

# Data

## Dictionaries

---

Notation    Meaning

$x = a + b$      $x$  consists of data element  $a$  &  $b$

$x = \{a/b\}$      $x$  consists of either  $a$  or  $b$

$x = (a)$      $x$  consists of an optional data element  $a$      $x =$

$y\{a\}$      $x$  consists of  $y$  or more occurrences     $x = \{a\}z$      $x$

consists of  $z$  or fewer occurrences of  $a$

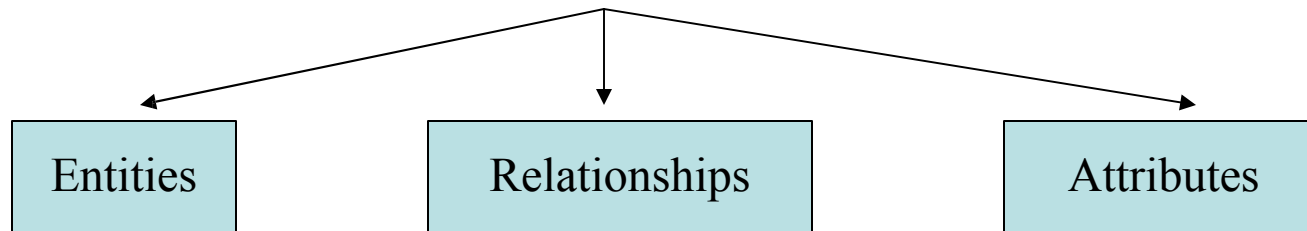
$x = y\{a\}z$      $x$  consists of between  $y$  &  $z$  occurrences of  $a$

# Entity-Relationship

## Diagrams

### Entity-Relationship Diagrams

It is a detailed logical representation of data for an organization and uses three main constructs.



### Entities

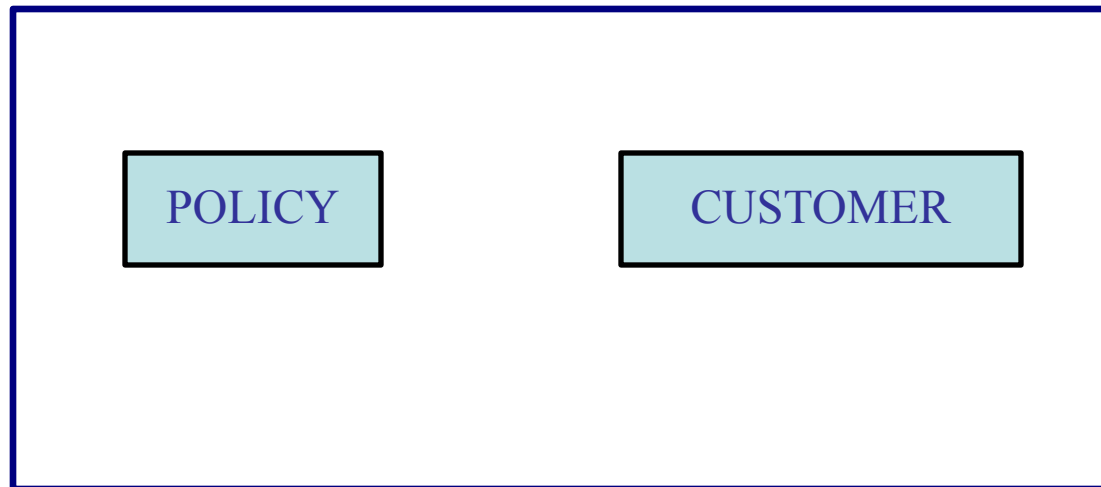
Fundamental thing about which data may be maintained. Each entity has its own identity.

Entity Type is the description of all entities to which a common definition and common relationships and attributes apply.

# Entity-Relationship

## Diagrams

Consider an insurance company that offers both home and automobile insurance policies .These policies are offered to individuals and businesses.





# Entity-Relationship

## Diagrams

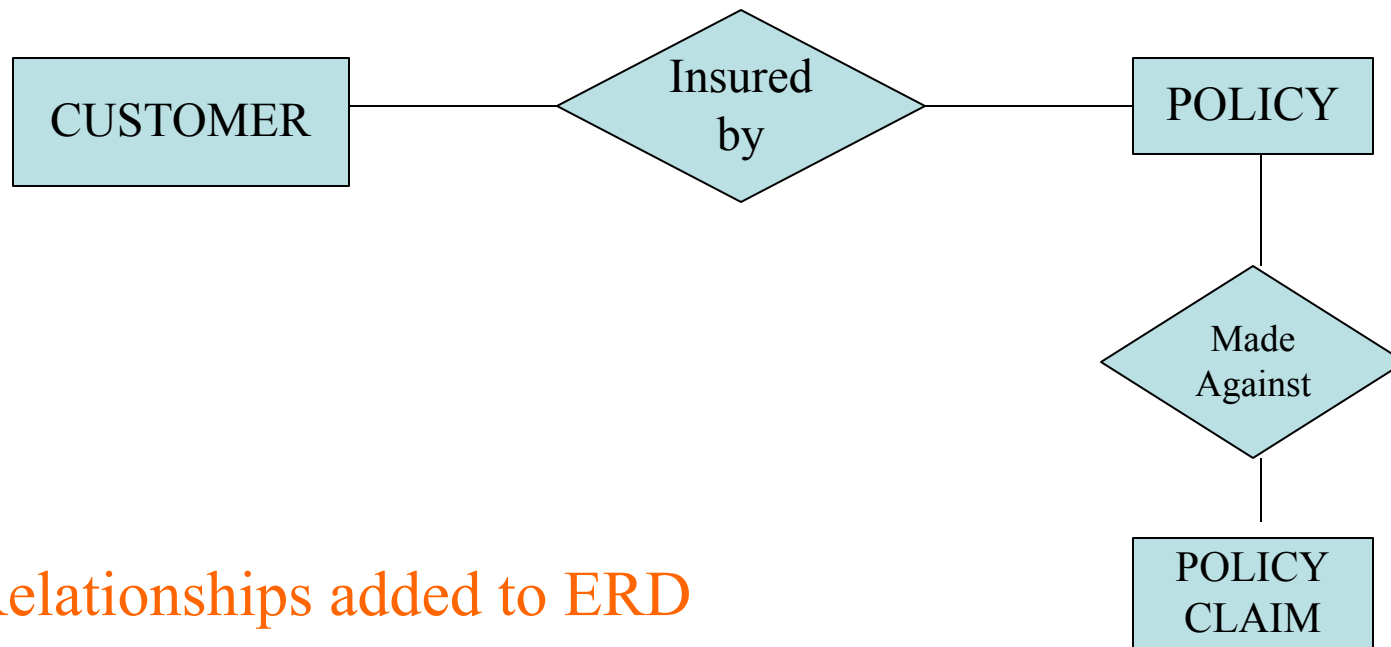
### Relationships

A relationship is a reason for associating two entity types.

Binary relationships involve ~~two~~ entity types

A CUSTOMER is insured by a POLICY. A POLICY CLAIM is made against a POLICY.

Relationships are represented by diamond notation in a ER diagram.

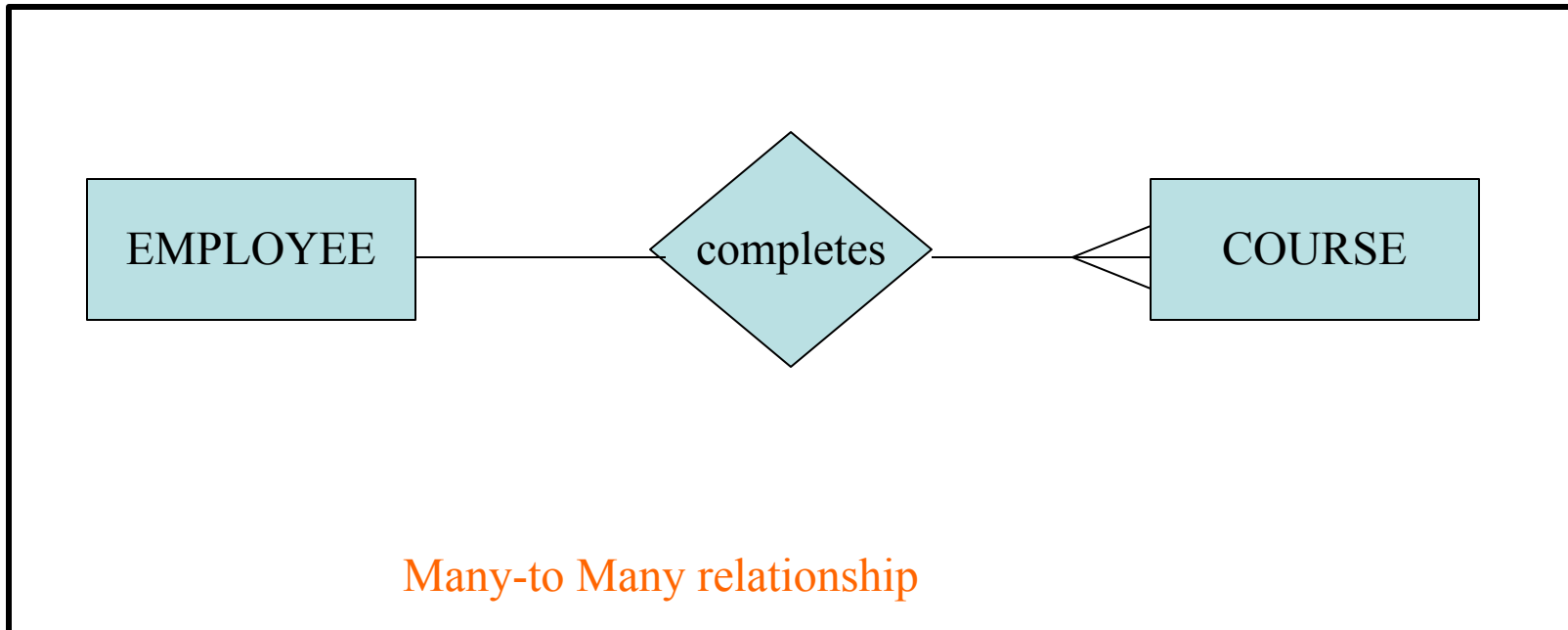


Relationships added to ERD

# Entity-Relationship

## Diagrams

A training department is interested in tracking which training courses each of its employee has completed.



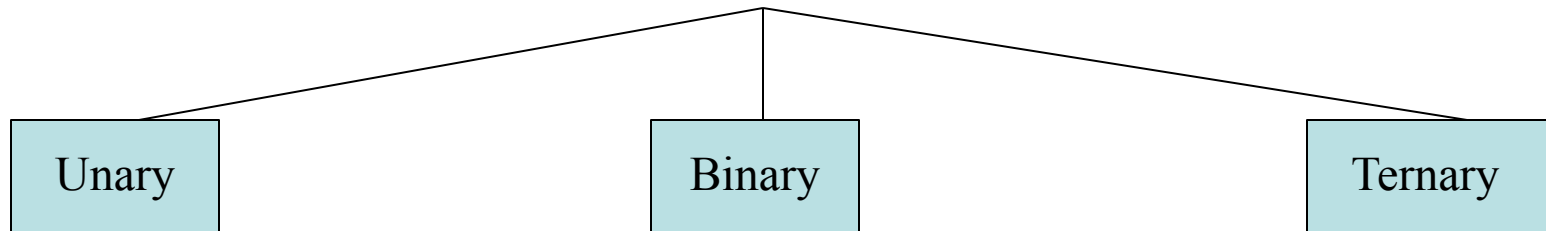
Each employee may complete more than one course, and each course may be completed by more than one employee.

# Entity-Relationship Diagrams

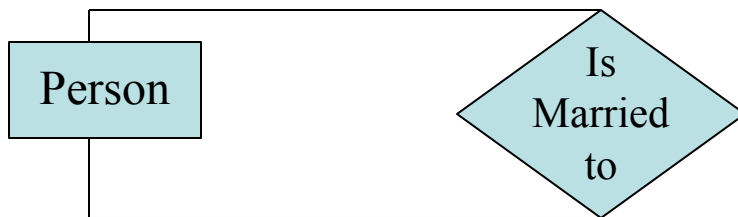
---

## Degree of relationship

It is the number of entity types that participates in that relationship.



### Unary relationship



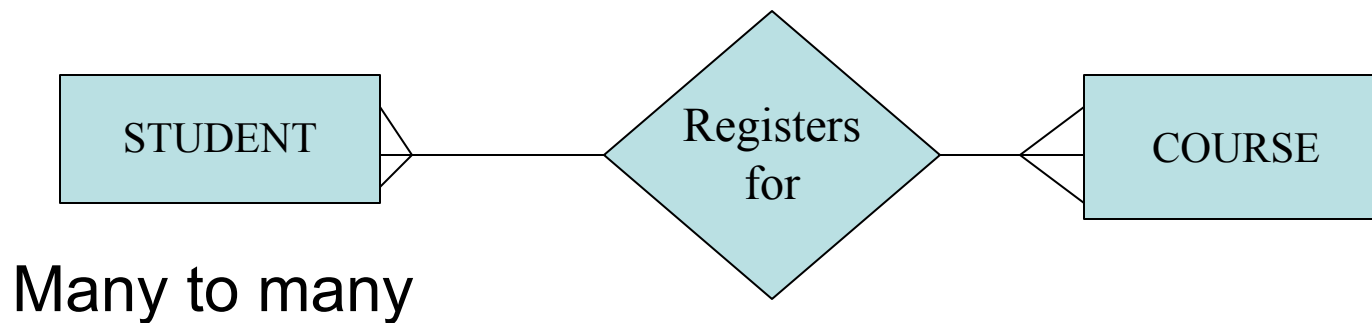
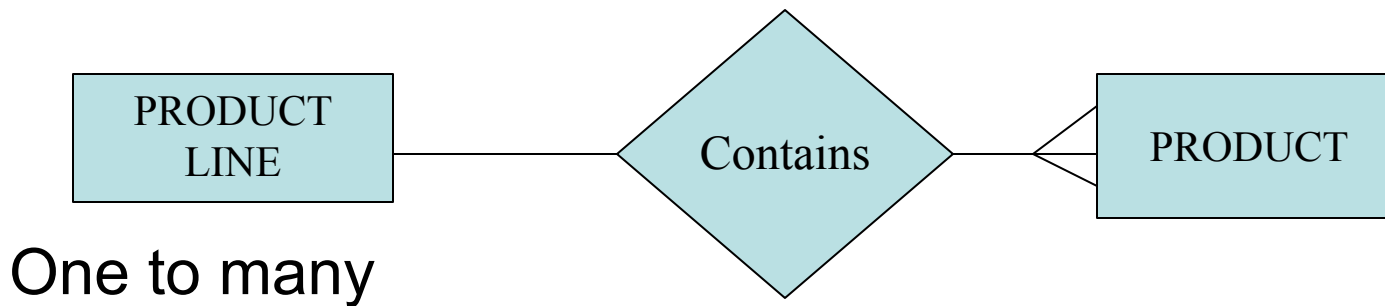
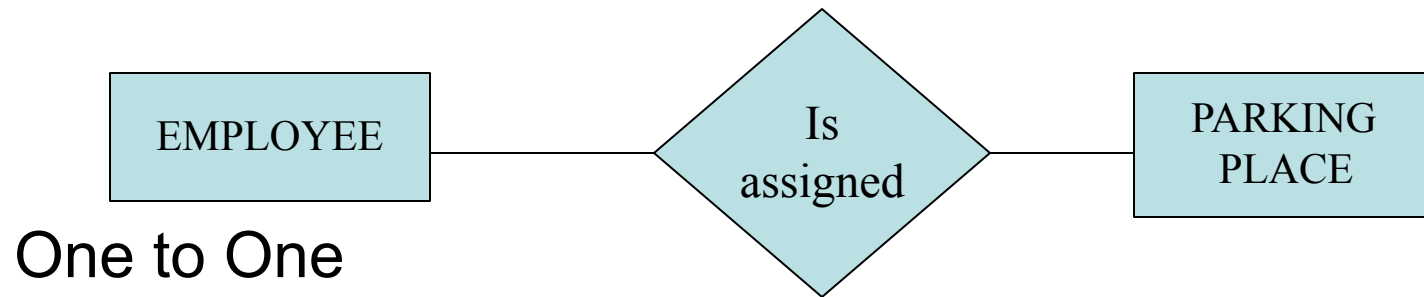
One to One



One to many

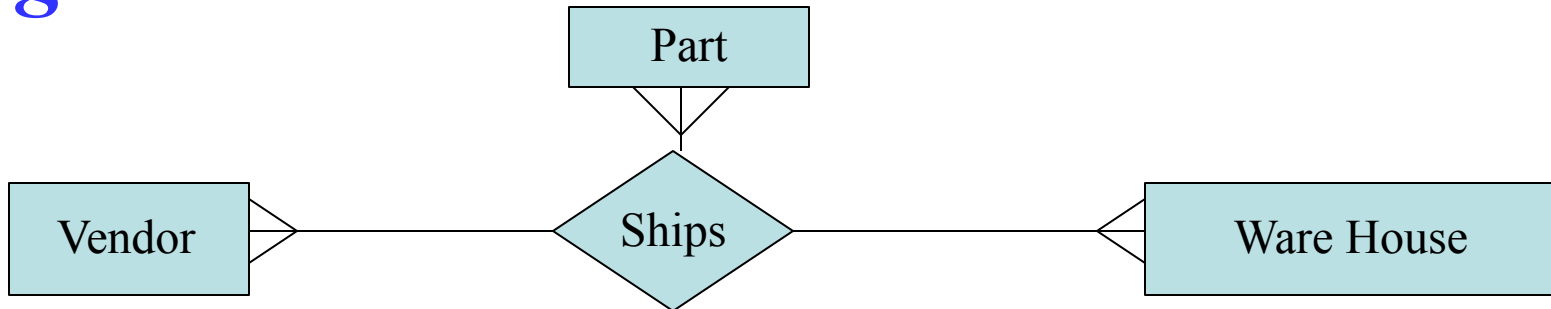
# Entity-Relationship

## Binary Relationship Diagrams



# Entity-Relationship

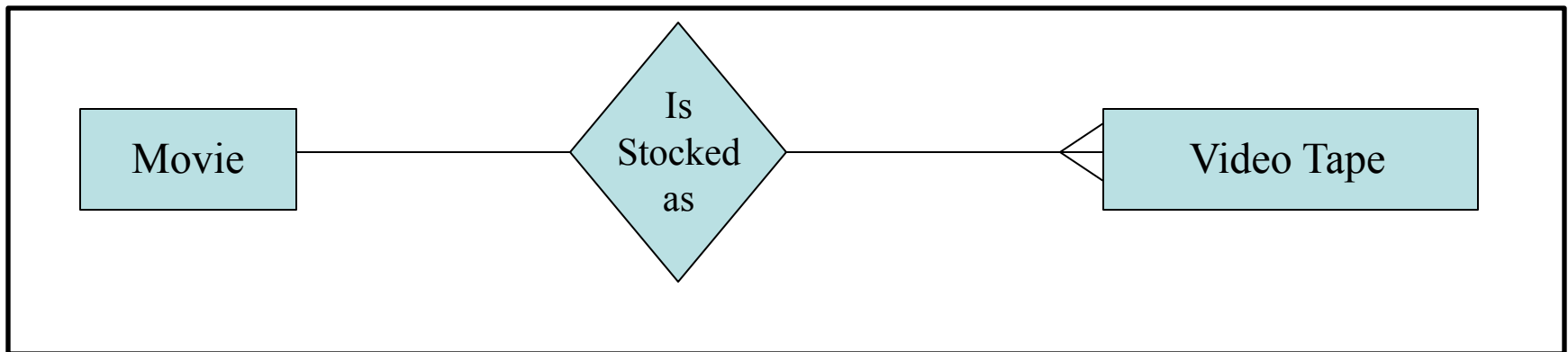
## Diagram Ternary relationship



## Cardinalities and optionality

Two entity types A,B, connected by a relationship.

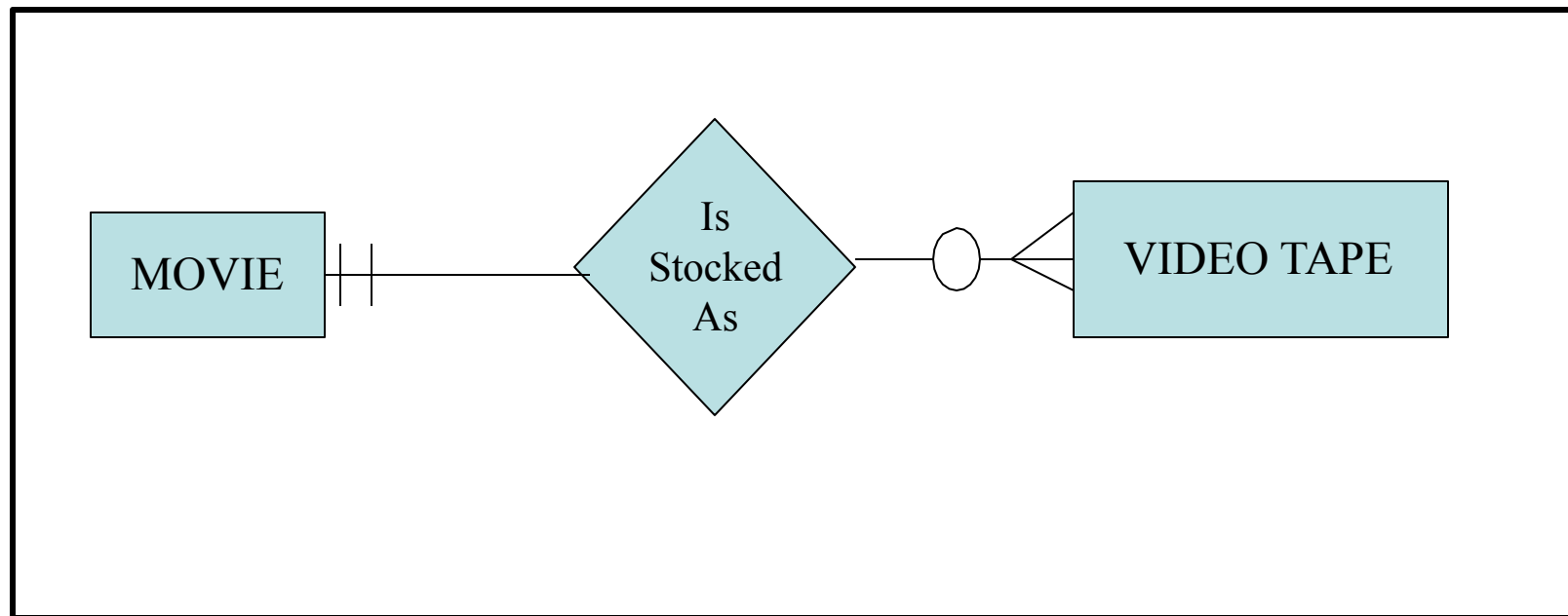
The cardinality of a relationship is the number of instances of entity B that can be associated with each instance of entity A



# Entity-Relationship

**Diagrams** Minimum cardinality is the minimum number of instances of entity B that may be associated with each instance of entity A.

Minimum no. of tapes available for a movie is zero. We say VIDEO TAPE is an optional participant in the is-stocked-as relationship.



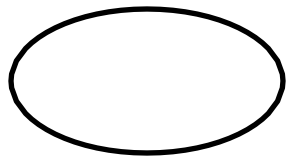
# *Entity-Relationship*

## *Diagrams*

### Attributes

Each entity type has a set of attributes associated with it.

An attribute is a property or characteristic of an entity that is of interest to organization.



Attribute

# *Entity-Relationship*

---

## *Diagrams*

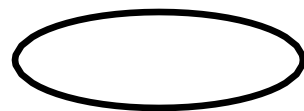
A candidate key is an attribute or combination of attributes that uniquely identifies each instance of an entity type.

Student\_ID       $\longrightarrow$       Candidate Key

If there are more candidate keys, one of the key may be chosen as the Identifier.

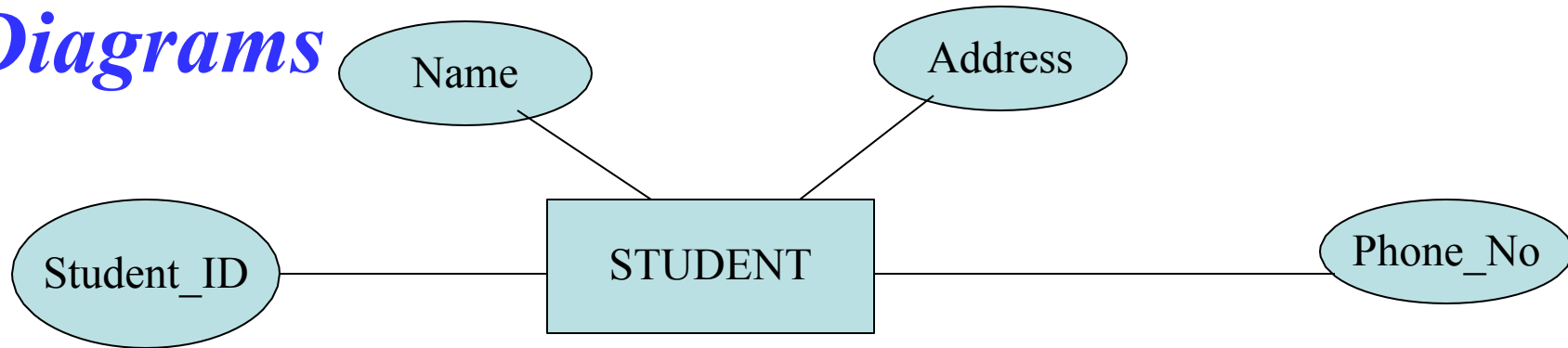
It is used as unique characteristic for an entity type.

Identifier

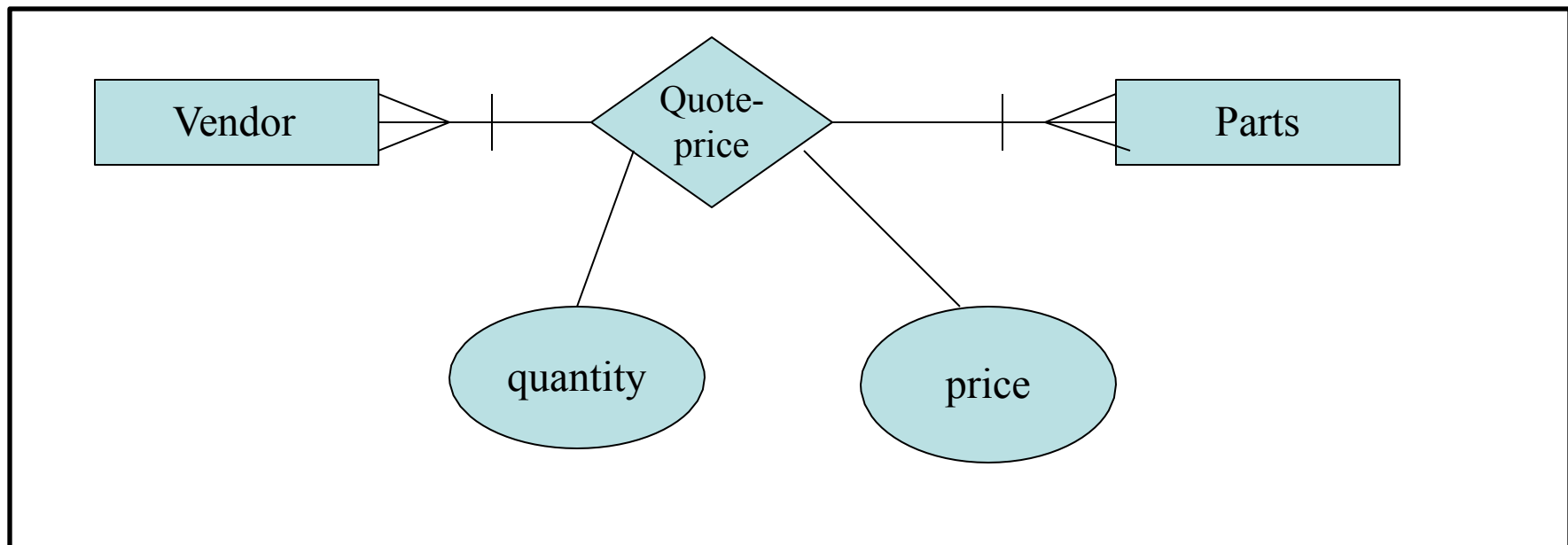




# *Entity-Relationship Diagrams*



Vendors quote prices for several parts along with quantity of parts.  
Draw an E-R diagram.



# *Approaches to problem analysis*

---

1. List all inputs, outputs and functions.
2. List all functions and then list all inputs and outputs associated with each function.

## Structured requirements definition (SRD)

### Step1

Define a user level DFD. Record the inputs and outputs for each individual in a DFD.

### Step2

Define a combined user level DFD.

### Step3

Define application level DFD.

### Step4

Define application level functions.

# Requirements

## Documentation

---

This is the way of representing requirements in a consistent format

SRS serves many purpose depending upon who is writing it.

- written by customer
  - written by developer
- 

Serves as contract between customer & developer.

# *Requirements*

## *Documentation*

---

### Nature of SRS

#### Basic Issues

- Functionality
- External Interfaces
- Performance
- Attributes
- Design constraints imposed on an Implementation

# *Requirements Documentation*

---

SRS Should

- Correctly define all requirements
- not describe any design details
- not impose any additional

constraints **Characteristics of a good SRS**

An SRS Should be

- ✓ **Correct**
- ✓ Unambiguous
- ✓ **Complete**
- ✓ **Consistent**

# *Requirements*

---

## *Documentation*

- ✓ Ranked for important and/or stability
- ✓ Verifiable
- ✓ Modifiable
- ✓ Traceable

# Requirements

---

## ~~Correct~~ Documentation

An SRS is correct if and only if every requirement stated therein is one that the software shall meet.

## Unambiguous

An SRS is unambiguous if and only if, every requirement stated therein has only one interpretation.

## Complete

An SRS is complete if and only if, it includes the following elements

- (i) All significant requirements, whether related to functionality, performance, design constraints, attributes or external interfaces.

# Requirements

---

(ii) Responses to both valid & invalid inputs.

(iii) Full Label and references to all figures, tables and diagrams in the SRS and definition of all terms and units of measure.

## Consistent

An SRS is consistent if and only if, no subset of individual requirements described in it conflict.

## Ranked for importance and/or Stability

If an identifier is attached to every requirement to indicate either the importance or stability of that particular requirement.



# *Requirements*

## *Documentation*

---

### Verifiable

An SRS is verifiable, if and only if, every requirement stated therein is verifiable.

### Modifiable

An SRS is modifiable, if and only if, its structure and style are such that any changes to the requirements can be made easily, completely, and consistently while retaining structure and style.

### Traceable

An SRS is traceable, if the origin of each of the requirements is clear and if it facilitates the referencing of each requirement in future development or enhancement documentation.

# *Requirements*

## *Documentation*

---

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                               |

# *Requirements Documentation*

---

## 2. The Overall Description

1. Product Perspective
  1. System Interfaces
  2. Interfaces
  3. Hardware Interfaces
  4. Software Interfaces
  5. Communication Interfaces
  6. Memory Constraints
  7. Operations
  8. Site Adaptation Requirements

# *Requirements*

## *Documentation*

---

2. Product Functions
3. User Characteristics
4. Constraints
5. Assumptions for dependencies
6. Apportioning of requirements

### 4. Specific Requirements

1. External Interfaces
2. Functions
3. Performance requirements
4. Logical database requirements
5. Design Constraints
6. Software System attributes
7. Organization of specific requirements
8. Additional Comments.

# *Requirements Validation*

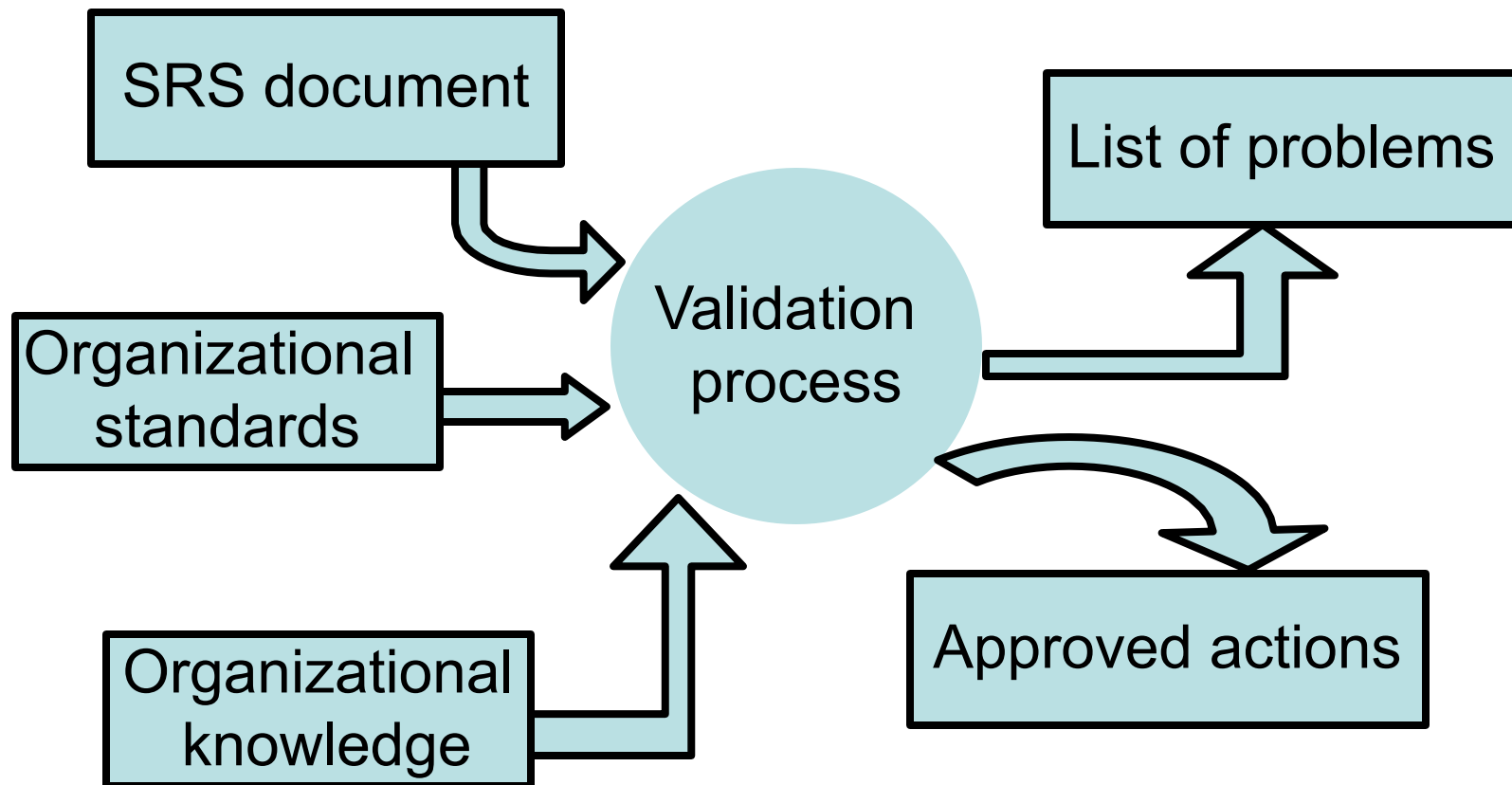
---

Check the document for:

- ✓ Completeness & consistency
- ✓ Conformance to standards
- ✓ Requirements conflicts
- ✓ Technical errors
- ✓ Ambiguous requirements

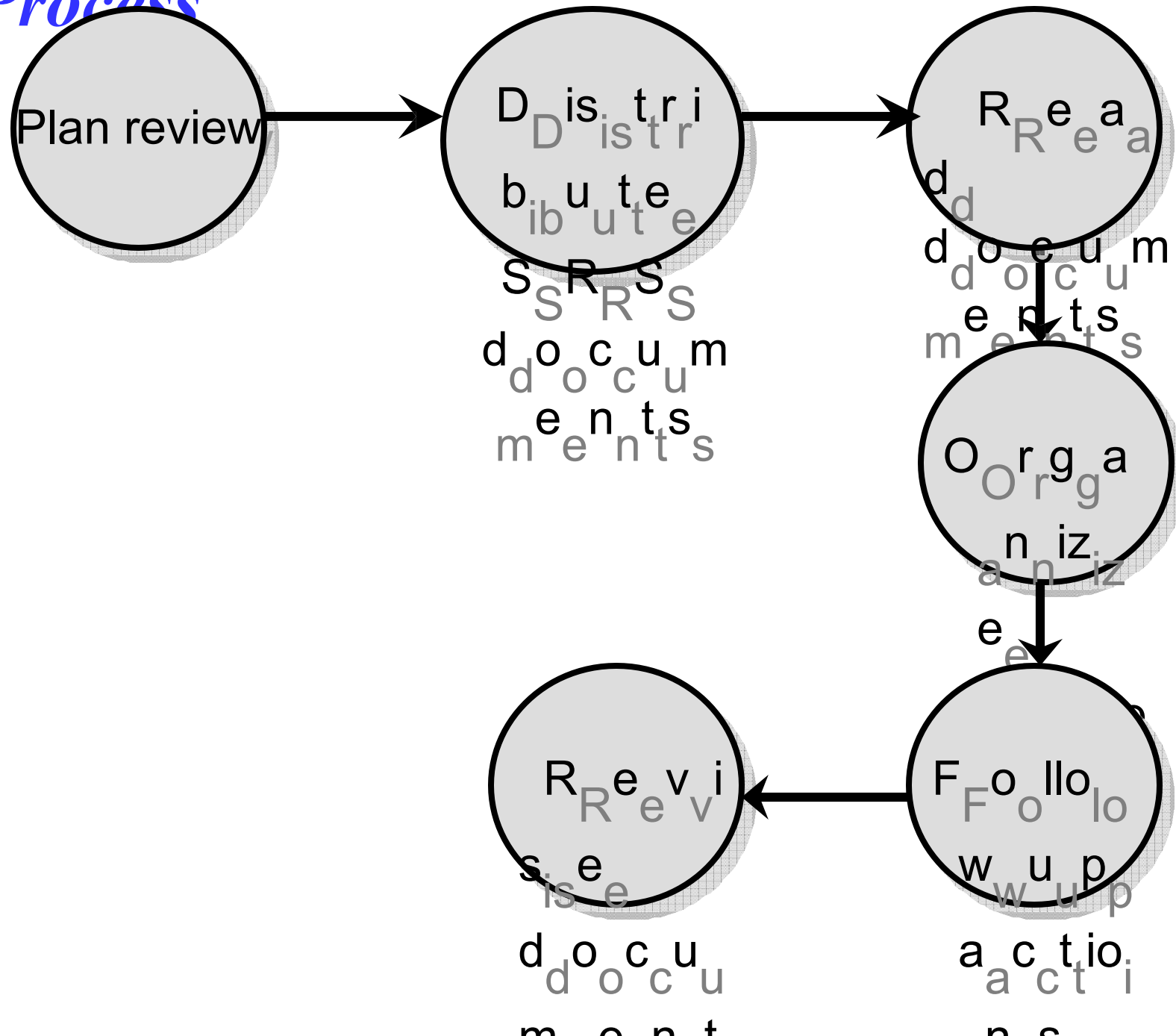
# *Requirements Validation*

---



# Requirements Review

## Process



# *Requirements Validation*

---

## **Problem actions**

- Requirements clarification
- Missing information
  - find this information from stakeholders
- Requirements conflicts
  - Stakeholders must negotiate to resolve this conflict
- Unrealistic requirements
  - Stakeholders must be consulted
- Security issues
  - Review the system in accordance to security standards



# *Review*

---

## *Checklists*

- ✓ Redundancy
- ✓ Completeness
- ✓ Ambiguity
- ✓ Consistency
- ✓ Organization
- ✓ Conformance
- ✓ Traceability

# *Prototypi*

---

## *ng*

Validation prototype should be reasonably complete & efficient & should be used as the required system.

# *Requirements Management*

---

- Process of understanding and controlling system requirements.

## ENDURING & VOLATILE REQUIREMENTS

- o Enduring requirements: They are core requirements & are related to main activity of the organization.

Example: issue/return of a book, cataloging etc.

- o Volatile requirements: likely to change during software development life cycle or after delivery of the product

# *Requirements Management*

---

## *Planning*

- Very critical.
- Important for the success of any project.

# *Requirements Change Management*

---

- Allocating adequate resources
- Analysis of requirements
- Documenting requirements
- Requirements traceability
- Establishing team communication
- Establishment of baseline