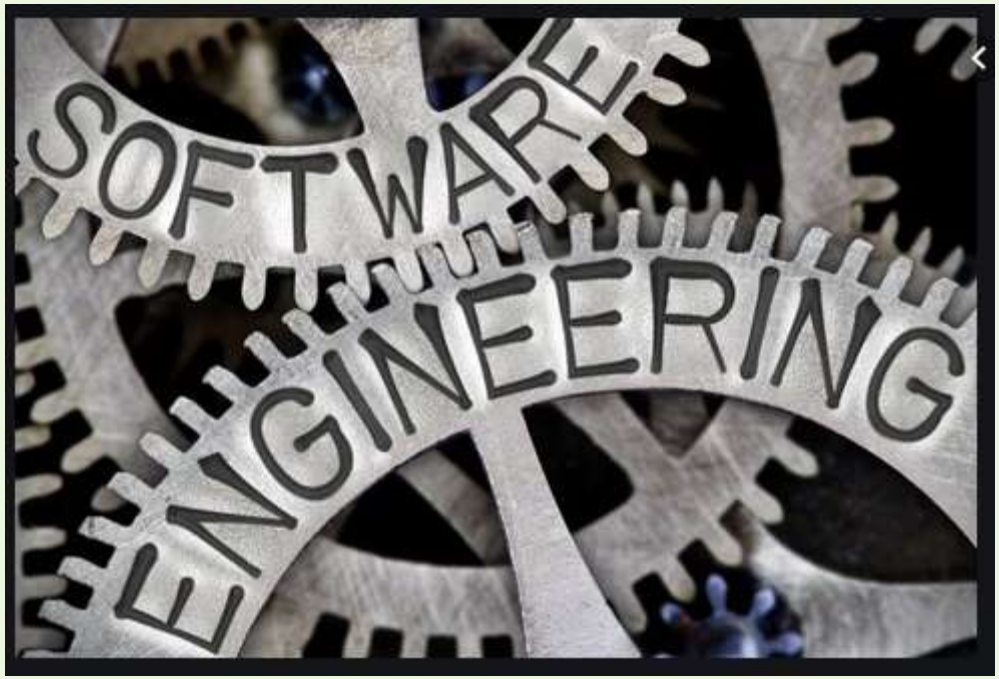
Software Engineering
Module-2



Faculty :

Dr. Pulak Sahoo

Associate Professor

Silicon Institute Of Technology

Text Books

(1) **Roger S. Pressman**, ***Software Engineering, A Practitioner's Approach***, Mc Graw Hill, 7th Edition, 2010

(2) **I Summerville**, ***Software Engineering,*** Pearson Education, 9th Edition, 2013

Reference Books

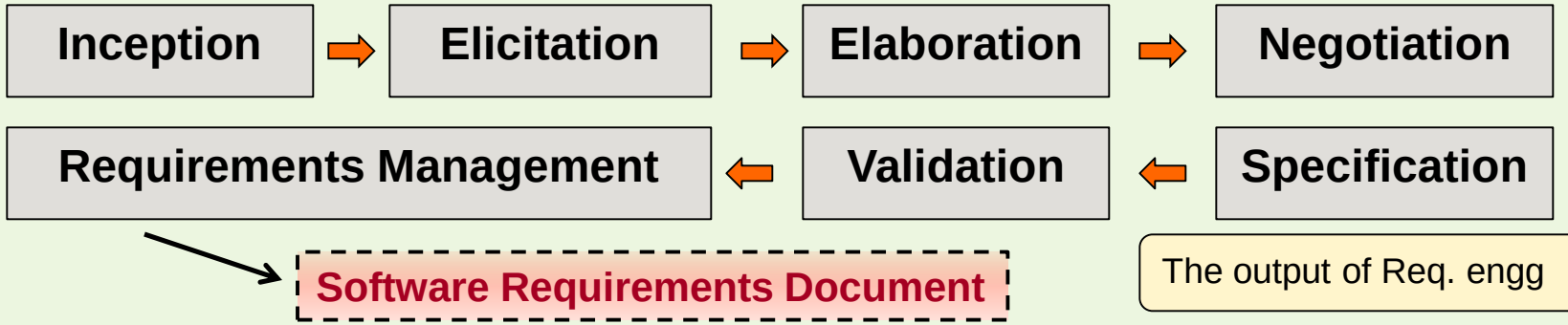
(1) **RAJIB MALL**, ***Fundamentals of Software Engineering***, PHI Learning, 4th Edition, 2014

Module2 Contents

- **Requirements Engineering**:
  - Types of Requirements, Functional and non-functional requirements
  - The software requirements document
  - Requirements - specification, engineering processes, elicitation & analysis, validation, and management
- **Decision Trees** & **Decision Tables**
- **Formal Specification**

Requirements Engineering

- The broad spectrum of **tasks** & **techniques** used to understand the **system requirements** is called ***requirements engineering***
- Begins during the **communication** activity & Continues into the **modeling** activity
- Must adapt to the needs of the process, the project, the product & the people involved
- **Requirements engg** builds a **bridge** to design & construction
- **Stages of Req. Engg. -** Inception, Elicitation, Elaboration, Negotiation, Specification, Validation & Requirements Management



Stages of Requirements Engineering

1. Inception

- Most **projects begin** when a business need is identified
- **Stakeholders** from the business community define a business case
- **Rough feasibility analysis** is done
- Working description of the **project's scope** is created

2. Elicitation

- **Business goals** are established
- **Stakeholders** share their goals honestly
- **Prioritization mechanism** for goals is established
- Potential **architecture** to meet stakeholder goals is created
- **Scope problems** occur when system boundary is ill-defined
- The **requirements-gathering** is started in an organized manner

Stages of Requirements Engineering

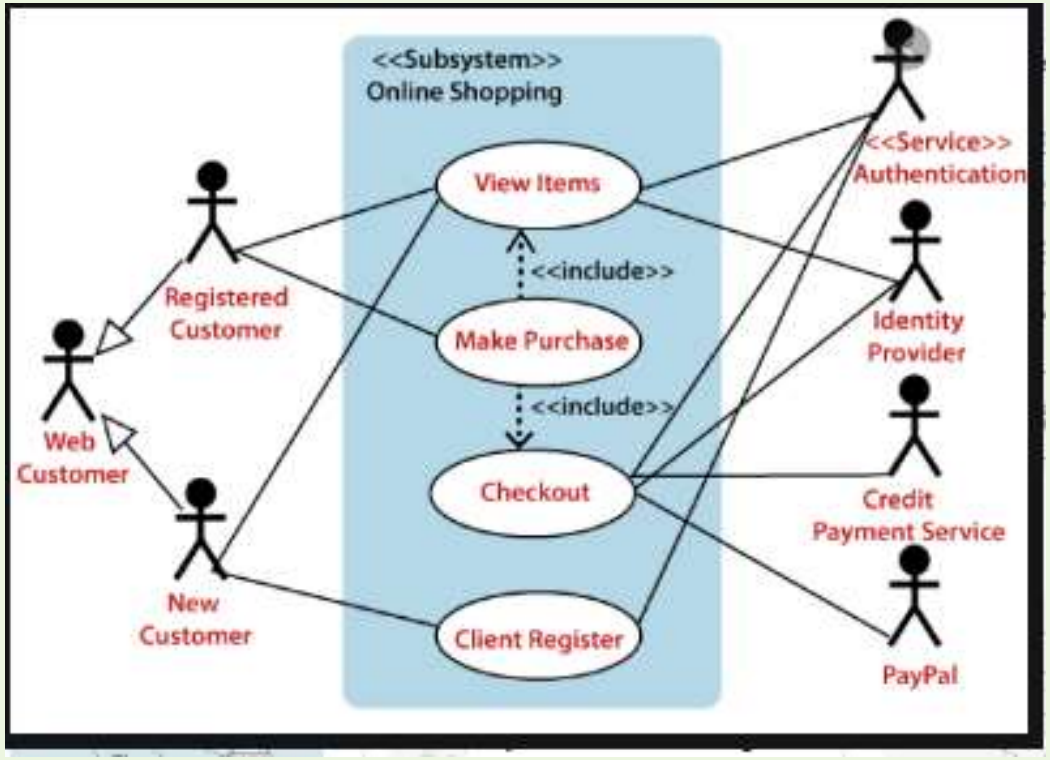
3. Elaboration

- Expansion & refinement of requirements model including various aspects of s/w function based on info obtained from customer
- Creation & refinement of user scenarios describing how the end user (actors) will interact with the system (use cases)
- Identification of relationships between classes & creation of a variety of supplementary diagrams *(class diagrams, activity diagrams etc..)*

4. Negotiation

- Customers sometimes ask for more than that can be achieved & propose conflicting requirements
- Reconcile these conflicts through a process of **negotiation**
- Discuss "conflicts in priority" & resolve them
- Achieve as much stakeholder satisfaction as possible

Use case diagram



Stages of Requirements Engineering

5. Specification

- Is a written document / a set of graphical models / a formal model / a collection of usage scenarios / a prototype / a combination of these
- A "Standard template" or a flexible format can used for a specification depending upon the need
- For large systems, a written document, combining natural language descriptions & graphical models may be used

6. Validation

- Requirements validation ensures that all s/w requirements are stated unambiguously
- Inconsistencies, omissions & errors have been detected & corrected
- A review team (including s/w engineers, customers, users & other stakeholders) does a "tech. review" of the requirements

Stages of Requirements Engineering

5. Requirements Management

- Requirements for computer-based systems change throughout the life of the system
- Requirements mgmt is a set of activities that help the project team identify, control & track requirements & changes to requirements as the project proceeds
- Stakeholders & s/w engineers work together on this like part of the same team

Requirements Analysis & Specification

Contents

- Introduction
- Requirements gathering & analysis
- Requirements specification
- SRS document
  - Different Sections in SRS
  - Good & Bad Properties of SRS
- Formal Specification

Requirements Analysis & Specification

- ✓ **Many projects fail:**
  - ✓ Because they start implementing the system:
    - ✓ Without clearly understanding **what the customer exactly wants**
- ✓ That's why it is important to learn:
  - ✓ *Requirements gathering, analysis & specification* techniques thoroughly

Requirements Analysis & Specification - activities

- **Consists of 3 main activities:**

1. Requirement Gathering

- Collection of all the data regarding the system to be developed

2. Requirement Analysis

- Remove all inconsistencies & anomalies from the requirements

3. Requirement Specification

- **Systematically organize** requirements into a Software Requirements Specification (SRS) document

1. Requirements Gathering

- If the project is to automate an existing system
  - **Ex**: *Automating existing manual payroll activities*
  - The task of the system analyst is a little **easier**
  - Analyst can immediately obtain:
    - Input & output formats
    - The operational procedures

Requirements Gathering (CONT.)

- In the **absence** of a working system
  - Lot of imagination & creativity are required to gather requirements from the scratch

- Interacting with the customer to gather relevant data through interviews, meetings, workshops, email/phone communications, surveys, questionnaires..

2. Analysis of Gathered Requirements

- **After gathering all the requirements, analyze it to:**
  - Clearly understand the user requirements
  - Detect *Inconsistencies, anomalies* & *incompleteness*
  - Resolve through *further discussions* with the customer

Anomaly

- An **anomaly** is an **ambiguity** in the requirement:
- **Examples:** In case of a "temperature control system"
  - Customer says turn off heater when temperature is very high

Inconsistent requirement

- Inconsistent requirement means some part of the requirement:
  - **Contradicts** with other parts

- **Example:** In case of a "temperature control system"
  - One customer says turn off heater and open water shower when temperature > 100 C
  - Another customer says turn off heater and turn ON cooler when temperature > 100 C

Incomplete requirement

- Some important parts of the requirements have been **omitted** :
  - due to oversight or
  - due to lack of clarity

- **Example**:
- In case of a "temperature control system"
  - The analyst has not recorded:
    when temperature falls below 90 C:
    - Heater should be turned ON
    - Water shower turned OFF

3. Software Requirements Specification

- Main aim of requirement specification:
  - Systematically organize the requirements
  - Document requirements properly

Software Requirements Specification

- The SRS document is useful in various contexts:

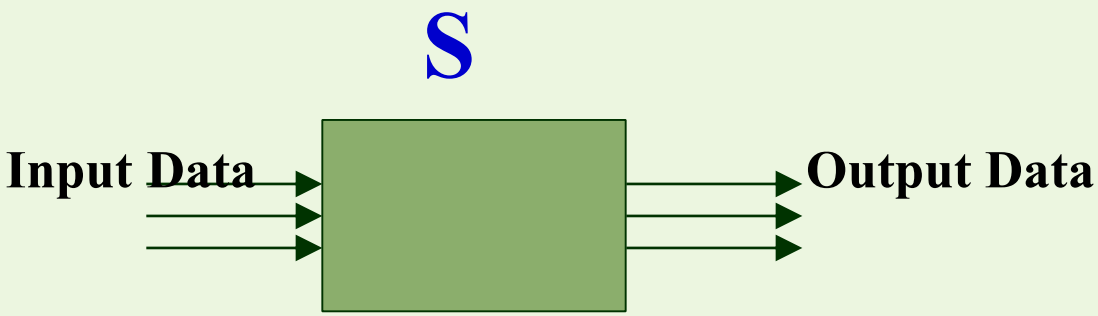
- It serves as:
  - Statement of user needs
  - Contract document
  - Reference document
  - Definition for implementation

Software Requirements Specification: A Contract Document

- SRS document is not only a ***reference*** document :
- It is also a contract between the development team and the customer
  - Once the SRS document is approved by the customer,
    - Any subsequent controversies are settled by *referring* the SRS document

SRS Document (CONT.)

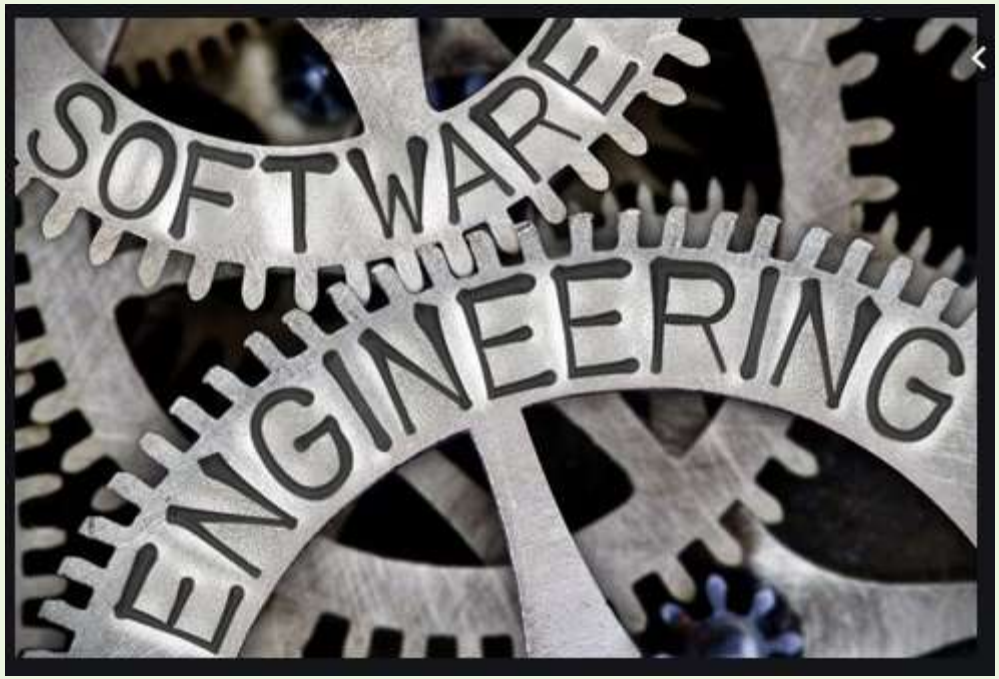
- The SRS document is known as black-box specification because :
  - It's *internal details are not documented*
  - Only its visible external behaviour (i.e. input/output) is documented



SRS Document (CONT.)

- SRS document concentrates on:
  - What needs to be done
  - Carefully avoids the solution ("how to do") aspects
- The requirements are documented:
  - Using end-user terminology

Software Engineering
Module-2



Faculty :

Dr. Pulak Sahoo

Associate Professor

Silicon Institute Of Technology

Module2 Contents

- **Requirements Engineering**:
  - Types of Requirements, Functional and non-functional requirements
  - The software requirements document
  - Requirements - specification, engineering processes, elicitation & analysis, validation, and management
- **Decision Trees** & **Decision Tables**
- **Formal Specification**

Properties of a good SRS doc

- It should be concise
  - At the same time should not be ambiguous
- It should specify what the system must do
  - Not say how to do it
- It should be well-structured & Easy to change
- It should be consistent & complete

- It should be traceable
  - One should be able to trace which part of the specification corresponds to which part of the design and code and vice versa
- It should be verifiable
  - **Ex.** "system should be user friendly" is not verifiable

Properties of Bad SRS Documents

- Unstructured Specifications:
  - Narrative essay one of the worst types of specification document:
    - Difficult to change, be precise, be unambiguous,
    - Has scope for contradictions

- Noise:
  - Presence of text containing information irrelevant to the problem
- Silence:
  - aspects important to proper solution of the problem are omitted
- Overspecification:
  - Addressing "how to" aspects
  - Over specification restricts the solution space
- Contradictions:
  - *Contradictions* might arise
    - if the same thing described at several places in different ways

- **Ambiguity:**
  - Unquantifiable aspects, e.g. "good user interface"
- **Forward References:**
  - References used earlier but defined only later on in the text
  - ***Ex**: using abbreviated terms like GUI & MIS etc. in the earlier parts of SRS but defining them later.*
- **Wishful Thinking:**
  - Descriptions of aspects for which realistic solutions will be hard to find
  - ***Ex**: The complex logic should be implemented in most simple manner or the user Interface should be the best.*

Main Components of SRS Document

- SRS document, mainly **contains** :
  - Introduction *(Problem statement in summarized form)*
  - Goals Of Implementation *(Describes the benefits offered to the stakeholders)*
  - Functional requirements *(Detailed description of each functional element of the system including inputs, outputs & processing)*
  - Non-functional requirements *(performance, interface, reusability, security, usability, maintainability etc..)*
    - Constraints on the system *(H/W, S/W, OS to be used, Standards compliance etc..)*

Functional Requirements

- Functional requirements describe:
  - A set of high-level requirements
  - Each high-level requirement:
    - Takes some input data from the user
    - Outputs some data to the user
  - Each high-level requirement:
    - Might consist of a set of identifiable functions

Functional Requirements

- For each high-level requirement:
  - Every **function** is described in terms of
    - input data set
    - output data set
    - processing required to get the output data set

Example Functional Requirements

- List all functional requirements with proper numbering
- **Req. 1: SEARCH BOOK**
  - User selects the "**search**" option,
    - he is asked to enter the key words
  - The system should output details of all books
    - whose title or author name matches any of the key words entered.
    - Details include: Title, Author Name, Publisher name, Year of Publication, ISBN Number, Catalogue Number, Location in the Library.

Example Functional Requirements

- Req. 2: **RENEW BOOK**
  - When the "**renew**" option is selected,
    - the user is asked to enter his *membership number and password*
  - After password validation,
    - the list of the books borrowed by him are *displayed*
  - The user can renew any of the books:
    - by clicking in the corresponding *renew box*

Non-functional Requirements

- Non-functional Requirements are those characteristics of the system which can not be expressed as functions:
- Some examples of NF requirements are
  - Performance
  - Portability
  - Security
  - Usability
  - Reusability
  - Reliability
  - Interface
  - Maintainability etc.
- *Examples of Performance, Reusability, Security etc can be given as per the SRS documents created in lab classes.*

Possible questions

Q: Classify the following as functional or non-functional requirement.

- Response Time of a Web Page **(Non- Functional)**
- Renew Book **(Functional)**
- Login **(Functional)**
- Authentication **(Non- Functional)**

Q: Using examples differentiate between functional and non-functional requirements.

- **Examples of F.R –** *in Library Info. System - Renew book, Create & cancel membership parts*
- **Examples of NF.R –** *in Railway Reservation System – Info. Inquiry &Ticket booking response time, authentication parts*

Constraints

- Constraints describe things that the *system should or should not do*
  - **Ex**
    - H/W, S/W, OS to be used
    - Standards compliance

Examples of constraints

- Hardware to be used
- Operating system or DBMS to be used
- Capabilities of I/O devices
- Standards compliance

Decision Trees
&
Decision Tables

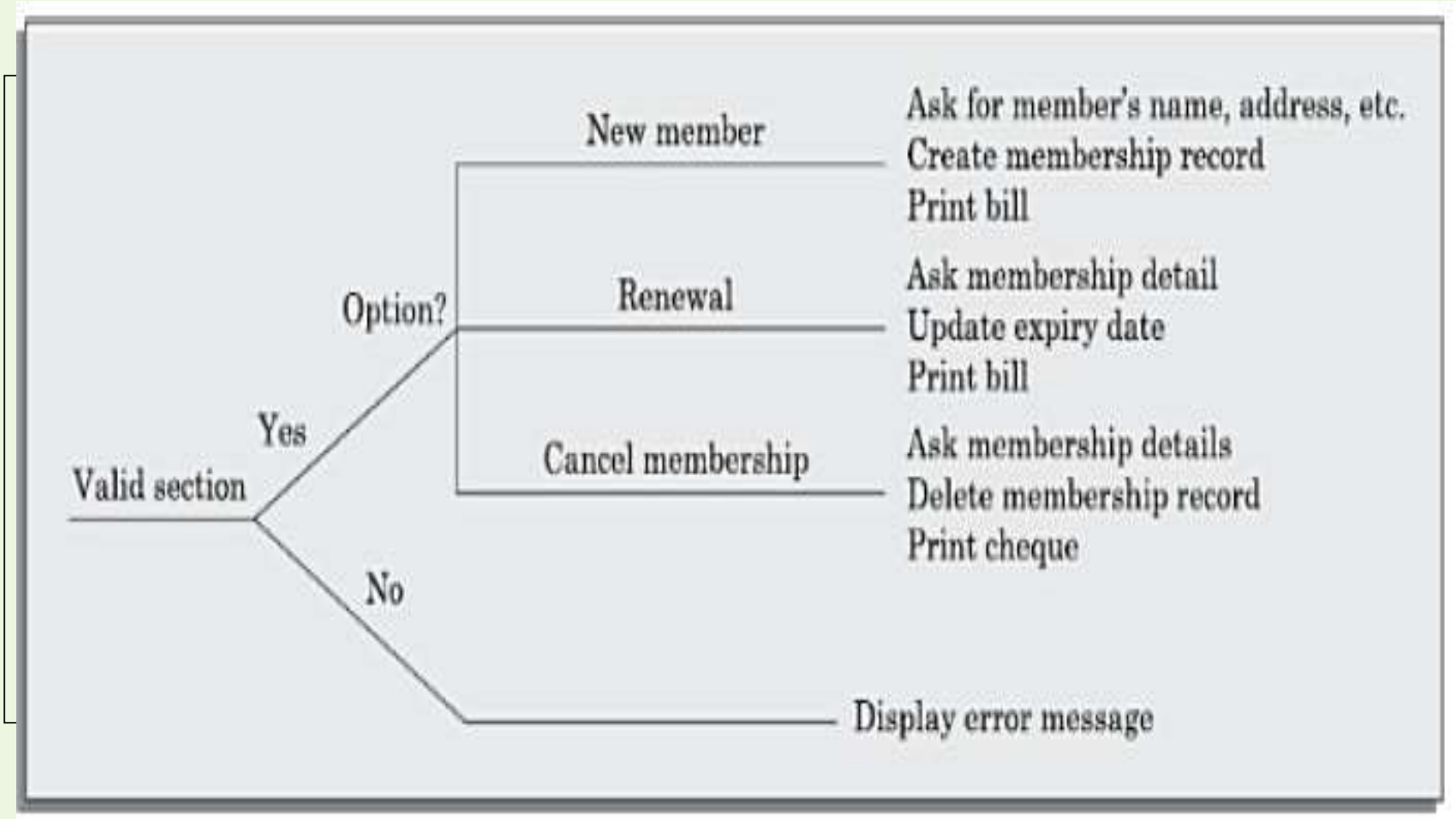
Decision tree & Decision table -
Techniques for Representing Complex Logic

- When the **Requirements** of the system are **complex** containing
  - Many different **scenarios**
  - **Condition** – **Decision** rules
- **Textual description** using natural languages may not be appropriate
- In such situations, a **decision tree** or a **decision table** can be used to represent the logic & the processing involved

Decision Tree

- **Decision Tree** gives a **graphic view** of :
  - The **processing logic** involved in **decision making** &
  - The corresponding **actions** to be taken
- The **edges** of a **Decision Tree** represent **conditions**
- The **leaf nodes** represent the **actions** to be performed
- ***Example*** *of DT for library membership management software (LMS) is given in the next page*

Decision Tree for LMS system



Decision Table

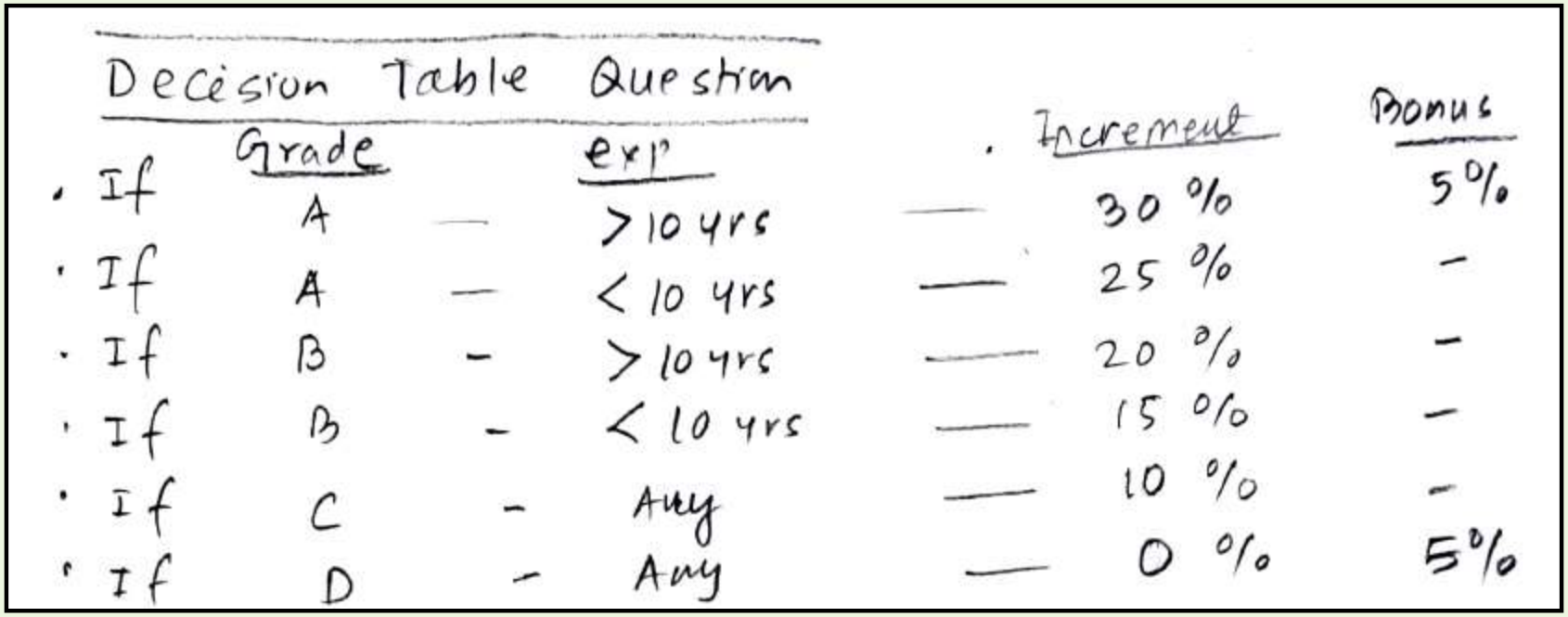
- Shows the **decision making logic** & the **corresponding actions** in a **tabular form**
- The **upper rows** of the table specify the **variables** or **conditions** to be evaluated
- The **lower rows** specify the **actions** to be taken
- A **column** in the table is called a ***rule***
- ***Example*** *of DT for library membership management software (LMS) is given in the next page*

Decision Table for LMS system

Table 4.1: Decision Table for the LMS Problem

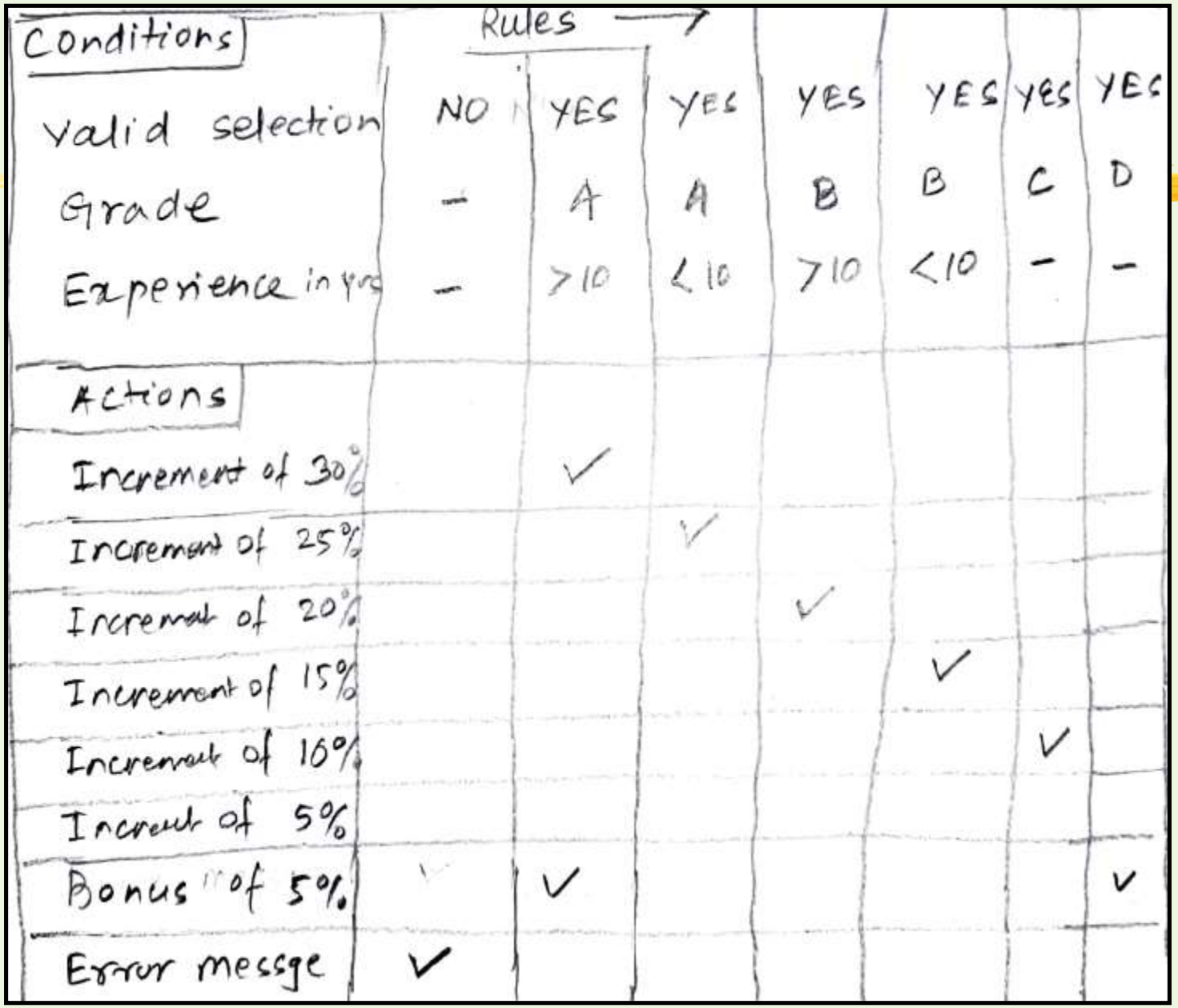
Conditions				
Valid selection	NO	YES	YES	YES
New member	-	YES	NO	NO
Renewal	-	NO	YES	NO
Cancellation	-	NO	NO	YES
Actions				
Display error message	×			
Ask member's name, etc.		×		
Build customer record		×		
Generate bill		×	×	
Ask membership details			×	×
Update expiry date			×	
Print cheque				×
Delete record				×

Ex2: Decision Table for Salary Increment



Decision Table Question

	Grade		exp		Increment	Bonus
• If	A	—	> 10 yrs	—	30 %	5%
• If	A	—	< 10 yrs	—	25 %	-
• If	B	-	> 10 yrs	—	20 %	-
• If	B	-	< 10 yrs	—	15 %	-
• If	C	-	Any	—	10 %	-
• If	D	-	Any	—	0 %	5%


Conditions
Rules →
Valid selection
NO
YES
YES
YES
YES
YES
YES
Grade
-
A
A
B
B
C
D
Experience in yrs
-
>10
<10
>10
<10
-
-
Actions
Increment of 30%
Increment of 25%
Increment of 20%
Increment of 15%
Increment of 10%
Increment of 5%
Bonus of 5%
Error messge

Possible Question

Q. Construct the Decision Table for the following business rules (L)

- The number of vacation days depends on age & years of service
- Every employee receives at least 22 days.
- Additional days are provided according to the following criteria:
- Only (employees < 18 yrs or at least 60 yrs, or employees with >= 30 years of service) will receive **5 extra days**
- (Employees with >= 30 years of service & employees of age >= 60 yrs) will receive **3 extra days**, on top of additional days already given.
- If (employee has >= 15 & < 30 years of service, **2 extra days** are given. These 2 days are also provided for employees of age 45 or more. These 2 extra days can not be combined with the extra 5 days.

Formal Specification

- A *formal specification technique* is a mathematical method to:
  - **Accurately specify** a system
  - **Verify** that *implementation satisfies specification*

R.Mall – page#260 onwards

Formal Specification

- Advantages:
  - Well-defined semantics, no scope for *ambiguity*
  - Automated tools can verify properties of specifications

Formal Specification

- Disadvantages of formal specification techniques:
  - Difficult to learn and use
  - Not able to handle complex systems

Formal Specification

- Mathematical techniques used include:
  - Axiomatic specification
  - Algebraic specification

(1) Axiomatic Specification

- In **Ax.S**, **First-Order Logic (FoL)** is used to: *(put more desc)*
- Write the **pre-** & **post- conditions** to specify the **operations** of the system in the form of **axioms**
- **Pre-conditions** - the conditions that must be satisfied before an operation can be invoked
- **Post-conditions** – the conditions that must be satisfied to consider the operation to be successful

  (Ex: constraints on the results produced)

Axiomatic Specification Ex-1

Example 1

Specify the pre- and post-conditions of a function that takes a real number as argument and returns half the input value if the input is less than or equal to 100, or else returns double the value.

$f(x : real) : real$

$pre : x \in R$

$post : \{(x \leq 100) \wedge (f(x) = x/2)\} \vee \{(x > 100) \wedge (f(x) = 2*x)\}$

Axiomatic Specification Ex-2

Input: *X[ ]* & *Key*
Output: *i* such that *X[i] = Key*

Example 2

Axiomatically specify a function named search which takes an integer array and an integer key value as its arguments and returns the index in the array where the key value is present.

search(X : IntArray, key : Integer) : Integer

pre : $\exists$ i $\in$ [Xfirst….Xlast], X[i] = key

post : {(X′[search(X, key)] = key) $\wedge$ (X = X′)}

Here, the convention that has been followed is that, if a function changes any of its input parameters, and if that parameter is named X, then it has been referred that after the function completes execution as X′.

Input parameter changed by function from *X [ ] to X ‘ [ ]*

Thanks!!!