SOFTWARE ENGINEERING

LECTURE NOTES

[UNIT – II]

* * *

PREPARED BY

Dr. J. Malla Reddy
Professor, Dept. of CSE

* * *

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
MAHAVEER INSTITUTE OF SCIENCE & TECHNOLOGY
(Affiliated to J.N.T. University, Hyderabad)
Keshavgiri(post), Bandlaguda, Hyderabad – 500 005

SEP, 2021

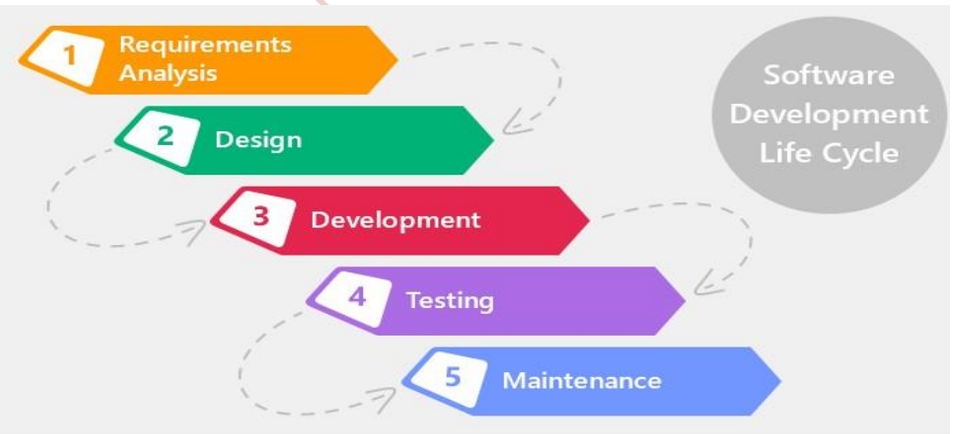
SOFTWARE ENGINEERING

Requirement Engineering is mechanism for understanding what the customer wants, analyzing the need, assessing the feasibility study, negotiating of a reasonable solution and specifying unambiguously transformed into operational system.

OBJECTIVES :

- Understand the concepts of user requirements and system requirements
- Understand the difference between functional & non function requirements.
- Describing system requirements using two techniques namely structured natural language and programming language .
- Understand how requirements may be organized in a software requirements document [SRS / SRD].

- **R**equirement Engineering is the crucial phase of Software Development Life Cycle[SDLC]
  - RE is contract between the client and developer.
  - The requirement engineering is most critical and complex socio technical multidisciplinary process of gathering requirements with social interaction .
  - Developer gather the requirements from the various viewpoints of the system
  - Gather the Complete and Consistent requirements.
  - The requirement have dominant impact on the software product.
  - RE crucial phase and reduce the errors at the early stage of software development.
  - Incomplete , imprecise, irrelevant requirement gathering can develop the poor quality and significant cost, time and efforts.



- The problems that software engineers have to solve are often immensely complex.
- Understanding the nature of the problems can be very difficult in case of system is new. Consequently , it is difficult to establish exactly what the system should do.
- The descriptions of the services and constraints are the requirements for the system and the process of finding out, analyzing documenting and checking these services and constraints is called "**Requirement Engineering ".**

- Once a contract has been awarded, the contractor must write a system definition for the client is more detail, so that the client understands and can validate what the software will do. Both of these documents may be called the requirement document of the system.

- Some of the problems that arise during the requirements engineering process are a result of failing to make a clear separation between these different levels of description.

USER & SYSTEM REQUIREMENTS :

- The **user requirements** to mean the high level abstract requirements.
- The **system requirements** to mean the detailed description of what the system should do.
- The detailed description is a bridge between requirement engineering and design activities.
- **User requirements** are statements, in a natural language plus diagrams, of what services the system is expected to provide and the constraints under which it must operate.
  - The **user requirements** should be written for client and contractor managers who don't have a detailed technical knowledge of the system.

- **System requirements** set out the system services and constraints in detail. The system requirement document , which is sometimes called a functional specification, should be precise. It may serve as a contract between the system buyer and software developer.
- The **system requirements** specifications should be targeted at senior technical staff and project managers. Again it will be used by staff from both the client and the contractor. System end-users may read both of these documents.
- A **software design specification** is an abstract description of the software design which is a basis for more detailed design and implementation. This specification what adds further detail to the system requirements specification.

CLASSIFICATION OF REQUIREMENTS

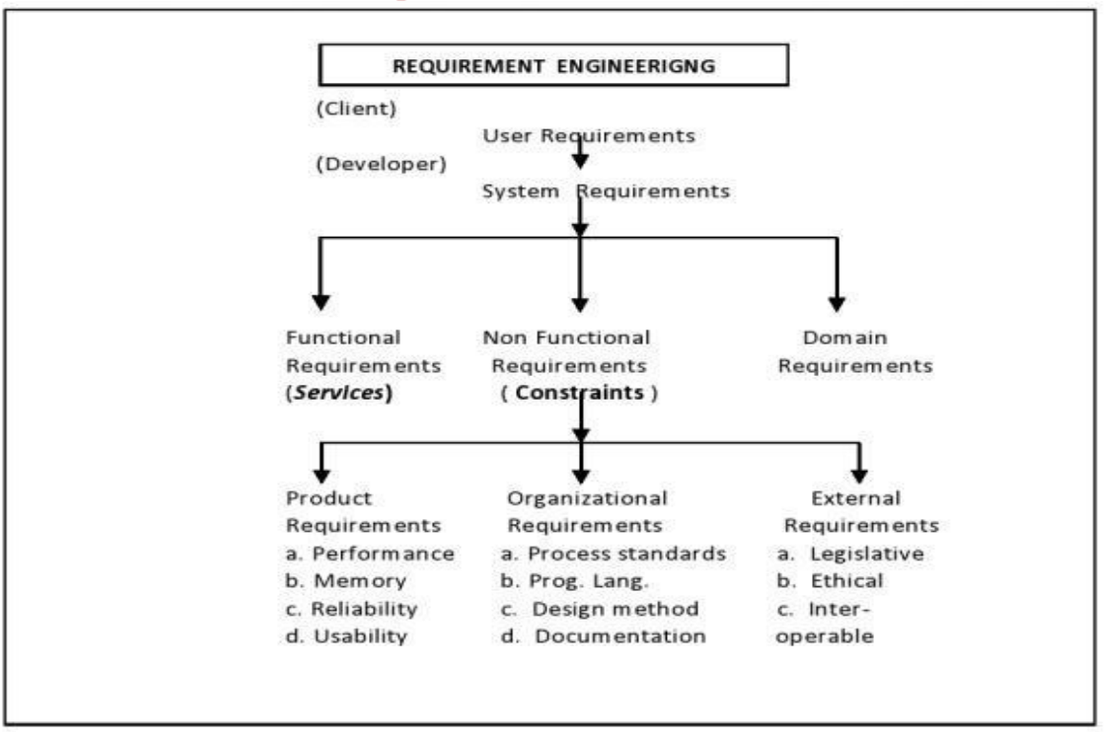


Fig. 1. Representation of Requirement Engineering.

Software system requirements are often classified as functional, non-functional requirements and domain requirements.

Functional requirements : These are statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particulars situations. In some cases, the functional requirements may explicitly state what the system should not do.

Non-functional requirements : These are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process, standards. Etc/

Domain requirements : These are requirements that come from the application domain characteristics of the system. They may be functional or non-functional requirements.

FUNCTIONAL REQUIREMENTS :

The functional Requirements for a system describe the functionality or services that the system is expected to provide. This depends on the type of software which is developed.

- When expressed as user requirements , they fairly in general way input, output, exceptions.
- The functional user requirements which define specific facilities which must be provided by the system.
  These are the requirements illustrated in the document (SRS).
- It is natural for a system developer to interpret unambiguous requirements to simplify the implementation
- New requirements have be established and changes made to the system. Of course , this delays system delivery increases costs.
- The functional requirements specification of a system should be both **complet**e and **consistent.** Completeness means that all services required by the user should be defined. The consistency means that requirements should not have contradictory definitions.
- In practice, for large, complex systems it is practically impossible to achieve requirements consistency and completeness.

NON FUNCITONAL REQUIREMENTS:

Non functional requirements are not directly concerned with the specific functions delivered by the system.

They may be related to emergent system properties such as reliability, response time and store occupancy. Alternately, they may define constraints on the system such as the capabilities of I/O Devices and the data representations used in the system interface.

Many Non functional requirements relate to the system as whole and individual system features.

- Failure to meet an individual functional requirements many degrade the system. But failure to meet an non functional requirements may make the whole system unusable.

* Non functional requirements may be constrain the process which may be used to develop the system.

- Non functional requirements arise through user needs, because of budget constraints, because of organizational policies, because of the needs of interoperable with other software or hardware systems or because of external factors such as regulations, privacy, legislation, etc.

TYPES OF NON FUNCTIONAL REQUIREMENTS

- **Product requirements** : These requirements that specify product behavior.
  (performance, memory, reliability (acceptable failure rate), portability, usability) .

- **Organizational requirements** : These are derived from policies and procedures in the customer's and developer organization.
  (process standards, implementation requirements such as Prog. Language, design method, delivery requirements of product and documentation).

- **External requirements** :This broad heading covers all requirements which are derived from factors external to the system and its development process.

These include interoperability requirements which define how the system interacts with system in other organization, legislative requirements, ethical requirements,

A common problem with non-functional requirements is that they are sometimes difficult to verify. They may be written to reflect general goals of the customer such as ease of use, the ability of the system to recover from failure or rapid user response.

Domain requirements : The Domain requirement are derived from the applications domain of the system rather than from the specific needs of system users. If these requirements are not satisfied , it may be impossible to make the system work satisfactorily.

PROBLEMS IN GATHERING REQUIREMENTS:

Problems :

Lack of clarity : With lack of clarity it is difficult to use language in precise and unambiguous . The document is wordy and difficult to read.

Requirement confusion : Functional and Non functional requirements system goals design information may not clearly distinguished.

Requirements amalgamation : Several different requirements may be expressed together as a single requirement.

It is good practice to separate user requirements from more detailed system requirements in the requirement document.

Otherwise the non-technical readers of user requirements may be overwhelmed by details which are really needed for technician.

SOFTWARE REQUIREMENT SPECIFICATION [SRS]:

* The SRS/SRD document is produced after identification of c**omplete** and **consistent** requirements.

* The software requirement specification is an effective **comprehensive description of behavior of the software** to be developed.

* The software requirement specification reduces **the time and effort** required by developers to achieve desired goals and also **minimizes the development cost**.

* **Fixing of errors are more easy** in the requirement document compare to later stages of development in terms of **cost and time.**

The effective SRS defines **how application** will interact with **system hardware , other programs and users** in a wide variety of real-world situations.

The requirement documentation represents various notations such a **Entity Relationship Diagrams, Data Flow Diagrams, Activity diagrams, State Transition Diagrams, Use-Case diagrams** are used to express the requirements at different levels in detail.

The requirement document represents the **functional and nonfunctional** requirements of the system.

Minimize misunderstanding :

- Write more detailed system requirement specification. Detailed for Designers, Developers Testers and support services. (precise, unambiguous)

- Invent the standard format and ensure that all requirement definitions adhere to that Format.
- Standardizing means – error omissions, easier check.
- Emboldening the initial requirements.
- Iterative process.
- Use language consistently. Distinguish between mandatory (Shell) and desired requirements (Should)

- Use test highlighting (Bold and italic) to pick out key parts of the requirement.

SOFTWARE REQUIREMENT SPECIFICATION [SRS]

1. Introduction

Purpose of the requirements document
Scope of the product
Definitions, acronyms and abbreviations
References
Overview of the remainder of the document.

2. General description

Product perspective
Product functions
User characteristics
General constraints
Assumptions and dependencies

3. Specific requirements cover functional, non-functional and interface requirements.

- Product Requirements.-
- Organizational Requirements.
- External Requirements.

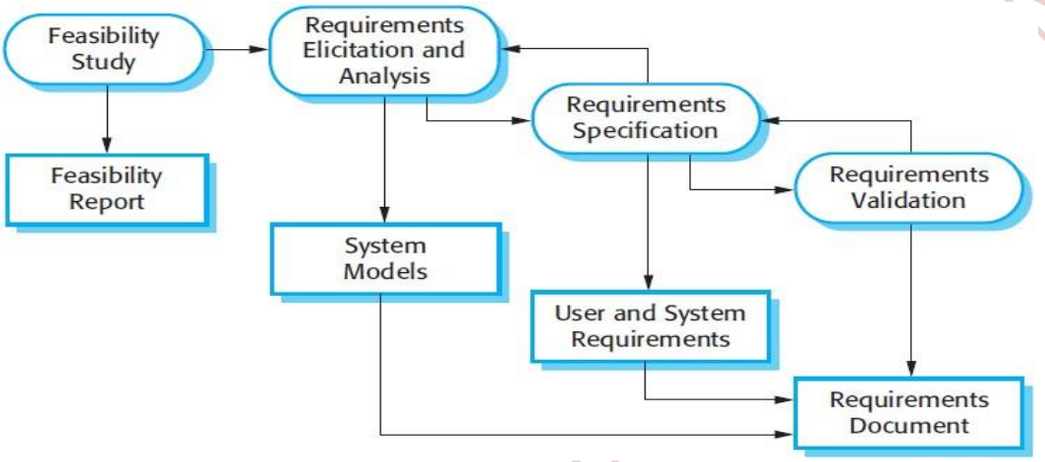
4. Appendices

5. Index

REQUIREMENT ENGINEERING PROCESS

OBJECTIVES:

- Understand the principal requirements engineering activities and their relationships.
- Discussion of several techniques of requirements elicitation and analysis.
- Understand the importance of requirement validation and how the requirements reviews are used.
- Necessity of requirement management how it supports other requirements engineering activities.



REQUIREMENT ENGINEERING PROCESS

Requirement Engineering is a process of that involves all the activities required to create and maintain a system requirement document.

- ***Feasibility study***
- ***Elicitation & Analysis of requirements. (Discovery of Requirements)***
- ***Specification of requirements and their documentation. (Structured format)***
- ***Validation of Requirements.***

Requirement Management which used to change the requirements for better understanding. (H/W, S/W , Organizational changes).

- Gready Booch defined better understanding system (Graphical System)

FEASIBILITY STUDY :

- Requirement Engineering starts with feasibility study
- Inputs to Feasibility study is an outline description of the system and how it is used within the organization.
- The feasibility study focused on various areas like behavioral, technical, economical and legal aspects of the development system of the following.
- The outcome of Feasibility Study indicates whether or not it is worthy system for development .
- Finally the requirement engineering team can verify *" What must be supported bye the system and what need not be supported*". Based on feasibility study, the requirement engineering team will go for the further development process

Feasibility is short focused study on:

- Does the new system can contributes overall objectives of the organization or elevating the problems in the place of old one.
- Is it possible to develop the new system using exiting technology and with in the cost and time constraints.
- Can the system be integrated with other systems which are already in place.
- Whether the new systems supports legal, ethical, legislative standards and procedures
- **" NO REAL VALUE "**

  (many organizations developed the systems which doesn't supports all objectives)

  Reasons :

  No clear statement of objectives (political / organizational factors, influence)

  Carrying out Feasibility Study involves (Information Assessment , information collection, Identify, Report writing)

Feasibility study Questions :

- How the organization cope up, if the system not implemented.
- What are the problems in current processes and how whould a new system help alleviate these problems.
- What are the direct contribution will the system make to the business objectives.
- Can information be transferred to and from other organizational systems.
- Does the system require technology which has not previously been used in the organization.
- What must be supported by the system and what need not be supported .

Sources : Managers of Departments, Engineers & Technical Experts , End users.

based on source of information, whether the system should do continue or to develop.

- It may propose scope, budget and schedule of the system .

REQUIREMENT ELICITATION & ANALYISIS : The requirement engineer mainly focuses on examining, gathering desired requirements/objectives for the system from different stakeholders**.**

- The requirement elicitation & Analysis is the next stage of RE-PROCESS.
- The process is in between the software development staff and customer end users.

Find out about

- About the application domain
- What are the services the system should provide
- The required performance of the system, hardware constraints and soon.
- It involves different kinds of people/ Stakeholders in the organization who effected with system (Direct/Indirect). (Managers, Engineers,Domain experts, Customers, Trade unions)

Problems in Elicitation & Analysis :

Stakeholders don't know that, what they actually wants from the developing system. They difficult to **articulate** requirement sometimes, requires **unrealistic** demands which are cost effective.

- Sometimes stakeholders express their requirements in their **own natural language with implicit knowledge.** The requirement engineer must understand the requirements.
- The **political factors** can influence the specific requirements of the system to **increase their influence and image.**
- Different stakeholder can express their **demands in different ways**. The requirement engineer can **identify the potential sources, commonalities & conflicts.**
- **Economic and business environment** in which the analysis take place is d**ynamic.**

REQUIREMENT ENGINEERING PROCESS ACTIVITIES

The various following process activities performed in the elicitation & analysis in iterative manner with continual feed back activity.

1. **Domain understanding**: Understanding problem and its definition.
2. **Requirement Collection**: Collection of stakeholders requirements from potential sources.
3. **Classification** : Similar types of requirements are group into coherent clusters.
4. **Conflict Resolution** : Eliminate conflicts between Requirements.
5. **Prioritization** : The requirement engineer prioritize requirements based on necessity.
6. **Requirement Checking** : Collect the complete requirements more consistent.

Requirements Discovery :

- Requirements discovery is the process of gathering information about the proposed and existing system and distilling the user and system requirements .
- Sources of information during the requirements discovery phase include documentation, system stakeholders and specifications of similar systems.
- The interaction can be made with stakeholders through interviews and observations and may use scenarios and prototypes, Ethnography to help with the requirements discovery.
- Example : [ATM]: Current bank customers, Representatives from other banks, Managers of bank branches, Database Administrators, Bank security mangers , The bank's marketing departments, H/W & S/W managers, National Banking regulators.

STAKEHOLDER VIEWPOINTS

The requirements sources (Stakeholders, domain, Systems) can all be represented as system viewpoints, where each viewpoints presents a sub set of the requirements for the system.

Viewpoints can be used as a way of classifying stakeholders and other sources of requirements.

- *Interactor viewpoints* : Interact directly with system.
  (Detailed system requirements,System features, interfaces)

- *Indirect viewpoints*: Represent stakeholders who do not use the system themselves but who influence the requirements in some way . (High level organizational Req/: , constraints)
- *Domain viewpoints :* Represent domain characteristics and constraints that influence the system requirements. (domain requirements apply for the system)

REQUIREMENT GATHERING METHODS

Interviews : Formal and informal interviews with system stakeholders are part of most requirement engineering process. Requirement Engg. Team put the questions to the stakeholders about the system to be developed. Interviews may be two types.

Closed Interviews : Predefined questions

Open Interviews : No predefined agenda.

In practice, interviews with stakeholders are normally mix of these.

Open ended discussions are rarely work well.

Difficulties & Advantages

Sceneries : Scenarios may be written as text, supplements by diagrams, screen shots and soon. Alternatively, a more structured approach such as event scenarios or use cases may be adopted.

Use cases : Use diagrams, Interaction diagrams.

Ethnography: Ethnography is an observational technique that can be used to understand social and organizational requirements. Satisfying the social and organizational requirements is often critical for the success of the system.

- Observations, day to day routines,
- Implicit requirements gathering

* Ethnography is particularly effective with following.
- Requirements that are derived from **cooperation and awareness** of other people's activities.

- Ethnography **can reveal critical process** details that are often missed by other requirements elicitation techniques.
-

REQUIREMENT VALIDATION

Requirement Validation & Verification:

- The requirement validation improves the quality of the requirement engineering process.
- Requirements are described in SRS, then all the stakeholders have to satisfy on its nature.
- Requirement engineer can ascertain the systems requirements against the raw requirements are called as "Requirement Validation" and verifying the correctness of System requirement documentation called as "Verification".
- The process of conformation of requirements in terms of unambiguous, complete, compatible and also testable for the further development process".
- Requirements validation is concerned with showing that the requirements actually defines the system that the customer wants. Requirements validation overlaps analysis is that it is concerned with finding problems with the requirements.

  Rework, Cost of fixing the errors

- During the requirements validation process checks should be carried out on the requirements in the requirements documents.
  - **Validity Checks**
  - **Consistency Checks**
  - **Completeness checks**
  - **Realism checks**
  - **Verifiability**

No. Of requirements validation techniques can be used in conjunction or individually.
(Requirements reviews, Prototyping, Test-case generation)

Requirements Reviews : Verifiability, Comprehensibility, Traceability, adaptability.

Requirement Management :

- In the **large scale systems changes are inevitable** with change of process and technology. Requirement management is the process of **managing the changes to requirements.**
- The process is performed **in parallel with other activities** in the software development.
- The activities of requirement management keeping **the development plan enhanced with new requirement**s, tracking and tracing the status of requirements.
- A requirement change can have **huge impact on the development process, which is very hard to estimate the cost and re-development work**.
- Requirement management tools can manage the **stable, and unstable requirements change and large volume of data** which are also collected during the this process.

SYSTEM MODELS

- Understand the importance the system boundaries and its context.
- Understand the concept of behavioral modeling , data modeling and object modeling.
- Introduction of notations defined in the UML and its implementation

- User requirements are in natural language which are understandable by the people who are not technical experts. However, more detailed systems requirements may expressed in more technical way

- Most widely used technique to describe the system specifications is the system models.

- These models are graphical representations described the business processes, the problem to be solved and the system to be developed.

- The system models are *bridge between* the analysis and design process.

- The models to be analyzed to develop an understanding of the existing system that to be replaced or improved or to specify the new system it is required.

System modelling helps the analyst to understand the functionality of the system and models are used to communicate with customers.

Different models present the system from different perspectives.

- **External perspective** showing the system's context or environment.
- **Behavioural perspective** showing the behaviour of the system.
- **Structural perspective** showing the system or data architecture.

- A system model is the abstraction of system being studied.
- A *representation* of a system should maintain all the information about the entity being represented.
- A *Abstraction* deliberately simplifies and picks out the most salient characteristics.

MODEL TYPES:

Different types of system models are based on different approaches of abstraction.

* **Data flow model** showing how the data is processed at different stages.
* **Composition model** showing how entities are composed of other entities.
- **Architectural model** showing principal sub-systems.
- **Classification model** showing how entities have common characteristics.
- **Stimulus/response model** showing the system's reaction to events.

UML is standard language for object oriented modeling defined by the Demarco, Rumbaughm, G. Boouch.

SYSTEM MODELS :

- Context models : Context Model

- Behavioural models: Data Flow Model, State Machine Model

- Data models: Entity, Entity Set, E-R Model, Semantic model, Data Dictionaries

- Object models : Inheritance, Object Aggregation, Object behaviour models

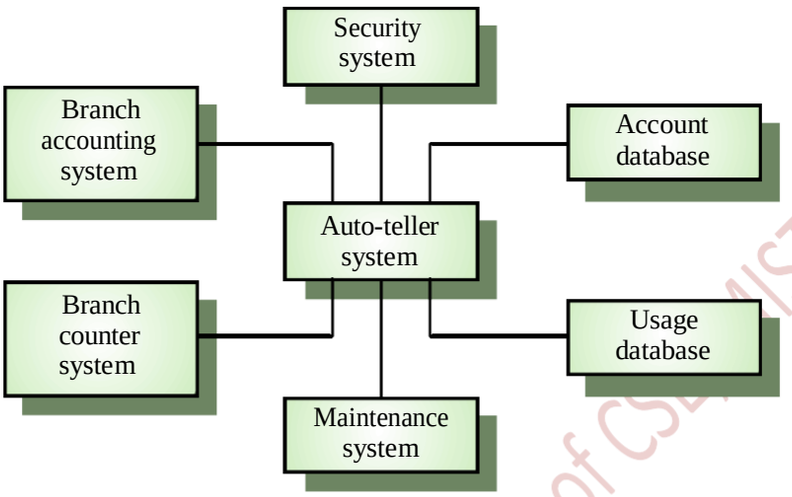
- Structured models : CASE workbenches

1. CONTEXT MODEL :

In the Elicitation Analysis stage -

Fix up the boundaries of the system. This involves working with system stake holders to distinguish what is the system and what is the system's environment. Decisions, system costs, time.

- The boundary between a system and its environment is relatively clear.



CONTEXT MODEL

Example : Library system (ex)

ATM (a/c, maintenance, database, security system, branch a./c system, branch counter)

Rectangles, lines, system and subsystems

- It shows the structure of the Information system, It will not show the relationship with other systems.

- Architectural models describe the environment of a system. It will not show the relation with other system existed in the environment. However the data to and fro with other systems.

2. BEHAVIOURAL MODELS :

- Behavioral Models are used to describe the overall behavior of the system.
  1. Data-flow models, 2. State machine models.

Data Flow Models : which model the data processing the system.
State machine models : which model how the system reacts to events.

These models may used separately or together depending on the type of the system that is being developed.

Data Flow Model :

The **dataflow models** may also be used to show the data that is transformed between the system and other systems in its environment.

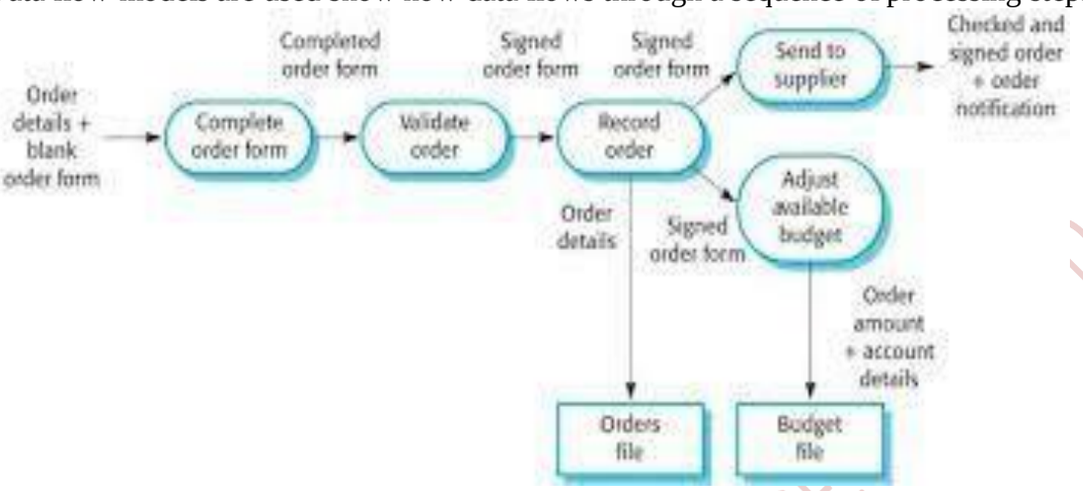
- **Data flow models** may be used to show **the processes and the flow of information** from one process to another

- Data flow models **show's how the data is processed** by the system.

- The widespread discussion made in the publications of ' Demarco' on structured analysis.

- The notation used in these models represents functional processing (**rounded rectangles**), data stores (**rectangles**) and data movement between function (**labeled arrows**).

- Data flow models are used show how data flows through a sequence of processing steps.



DATA FLOW MODEL

- In analysis model people, computer can carry the processing.
  A data flow model which shows the steps involved in processing an order the goods (such as computer equipment) in an organization)

- System and subsystem can exchange the information.

- Data flow models are valuable because **tracking and documenting how the data associated with a particular process moves through the system helps analysis understand** what is going on**.**

- The data flow models are functional perspective where each transformation represents a single function or process.

* Non technical factors can fix the boundaries of the system.

State Machine models :

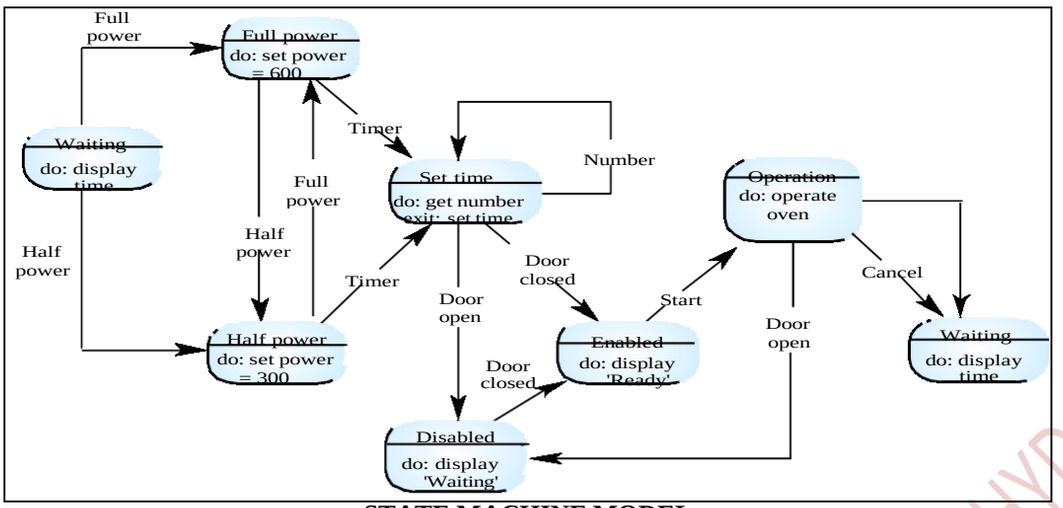
A **state machine models** describes how a system responds to **internal or external events**.

- The state machine model shows system and **events that cause transitions from one to another. It doesn't show the flow of the data within the system**.

- This type of model is often used for modeling **real-time systems** because these systems are often driven by **stimuli form the system environment**. Ex: the real-time alarm responds to stimuli from movement sensors, door-opening sensors and so on.
  Example : Micro Wave Oven

- State machine model are an **integral part of the real-time methods** proposed by the **Ward and Mellor** with state charts.

- The state charts are basis for the state machine modeling notations in UML.

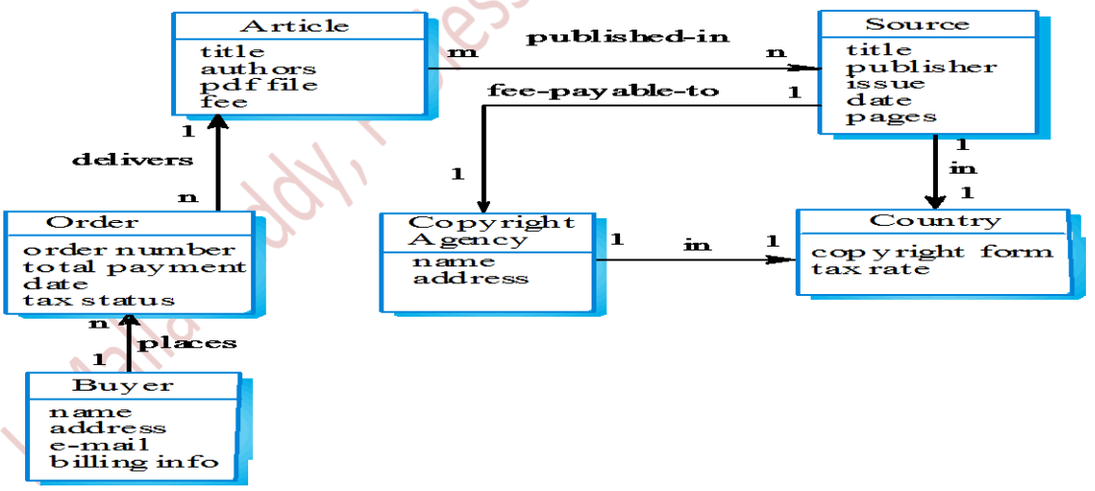


STATE MACHINE MODEL

- A state machine model of a system assumes that, **at any time, the system is in one of a no. of possible states**. When a stimulus is received, this may trigger a transaction to a different state.

3. DATA MODELS : Semantic data models :

- Used to describe the logical structure of data processed by the system.
- An entity-relation-attribute model sets out the entities in the system, the relationships between these entities and the entity attributes
- Widely used in database design. Can readily be implemented using relational databases.
- No specific notation provided in the UML but objects and associations can be used.



Library Semantic data models

Data dictionaries :

Name	Description	Type	Date
Article	Details of the published article that may be ordered by people using LIBSYS.	Entity	30.12.
authors	The names of the authors of the article who may be due a share of the fee.	Attribute	30.12.
Buyer	The person or organisation that orders a copy of the article.	Entity	30.12.
fee-payable-to	A 1:1 relationship between Article and the Copyright Agency who should be paid the copyright fee.	Relation	29.12.
Address (Buyer)	The address of the buyer. This is used to any paper billing information that is required.	Attribute	31.12.

4. OBJECT MODELS

- Object models describe the system in terms of object classes and their associations.
- An object class is an abstraction over a set of objects with common attributes and the services (operations) provided by each object.
- Various object models may be produced
  a. Inheritance models; b. Aggregation models; c. Object Behaviour models
- Natural ways of reflecting the real-world entities manipulated by the system.
- More abstract entities are more difficult to model using this approach.
- Object class identification is recognised as a difficult process requiring a deep understanding of the application domain.
- Object classes reflecting domain entities are reusable across systems.

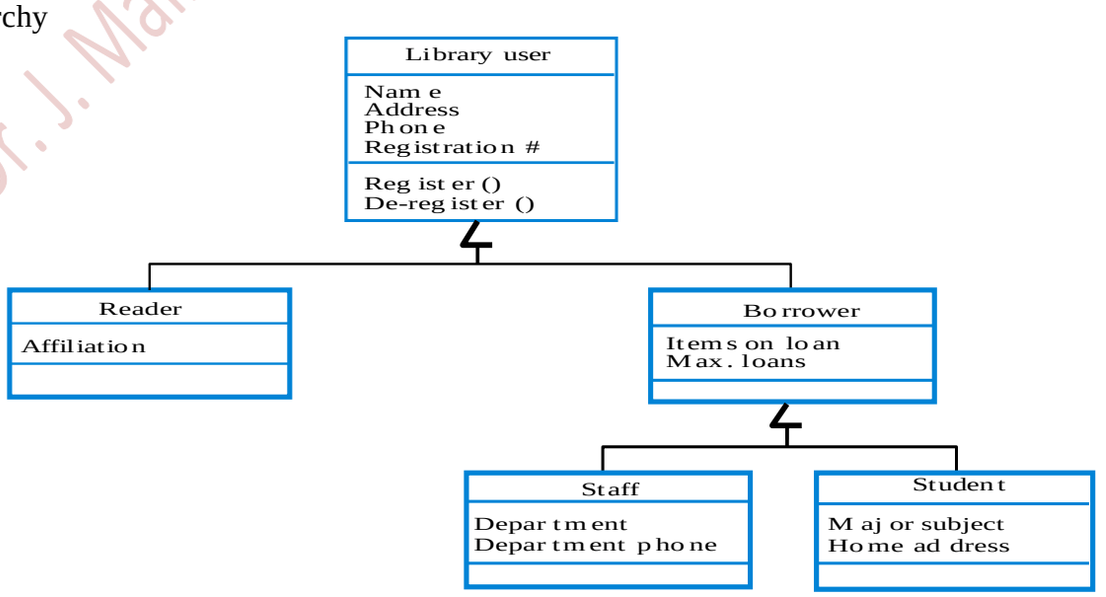
Inheritance models :

- Organise the domain object classes into a hierarchy.
- Classes at the top of the hierarchy reflect the common features of all classes.
- Object classes inherit their attributes and services from one or more super-classes. these may then be specialised as necessary.
- Class hierarchy design can be a difficult process if duplication in different branches is to be avoided.

Object models and the UML :

- The UML is a standard representation devised by the developers of widely used object-oriented analysis and design methods.
- It has become an effective standard for object-oriented modelling.
- **Notation**
  - Object classes are rectangles with the name at the top, attributes in the middle section and operations in the bottom section;
  - Relationships between object classes (known as associations) are shown as lines linking objects;

Inheritance is referred to as generalisation and is shown ‘upwards’ rather than ‘downwards’ in a hierarchy



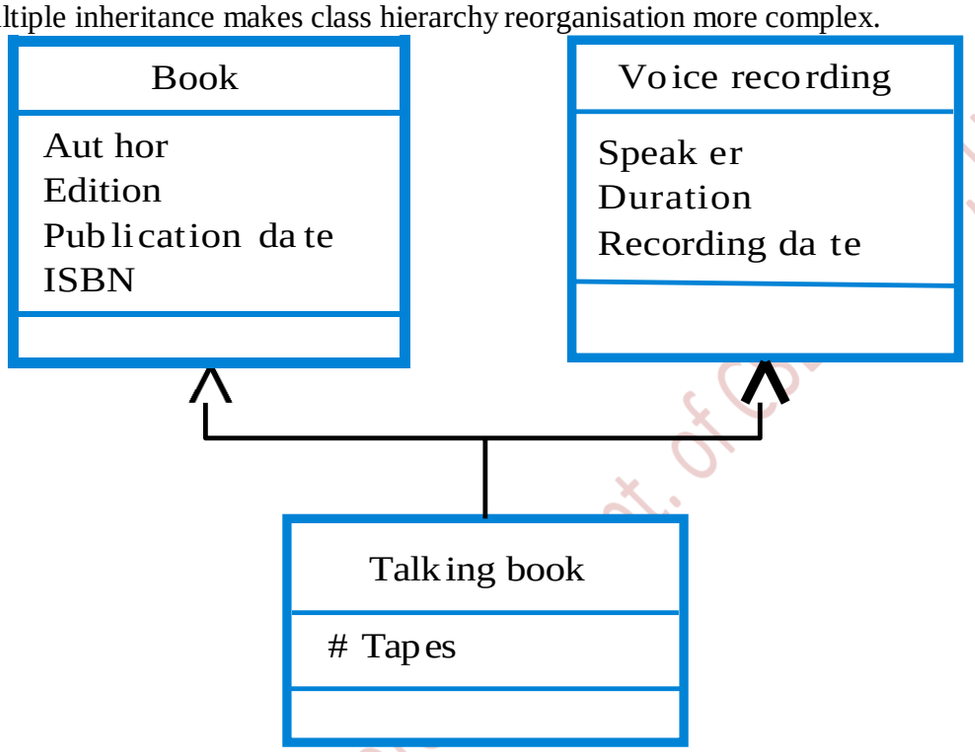
User class hierarchy

Multiple Inheritance :

- Rather than inheriting the attributes and services from a single parent class, a system which supports multiple inheritance allows object classes to inherit from several super-classes.

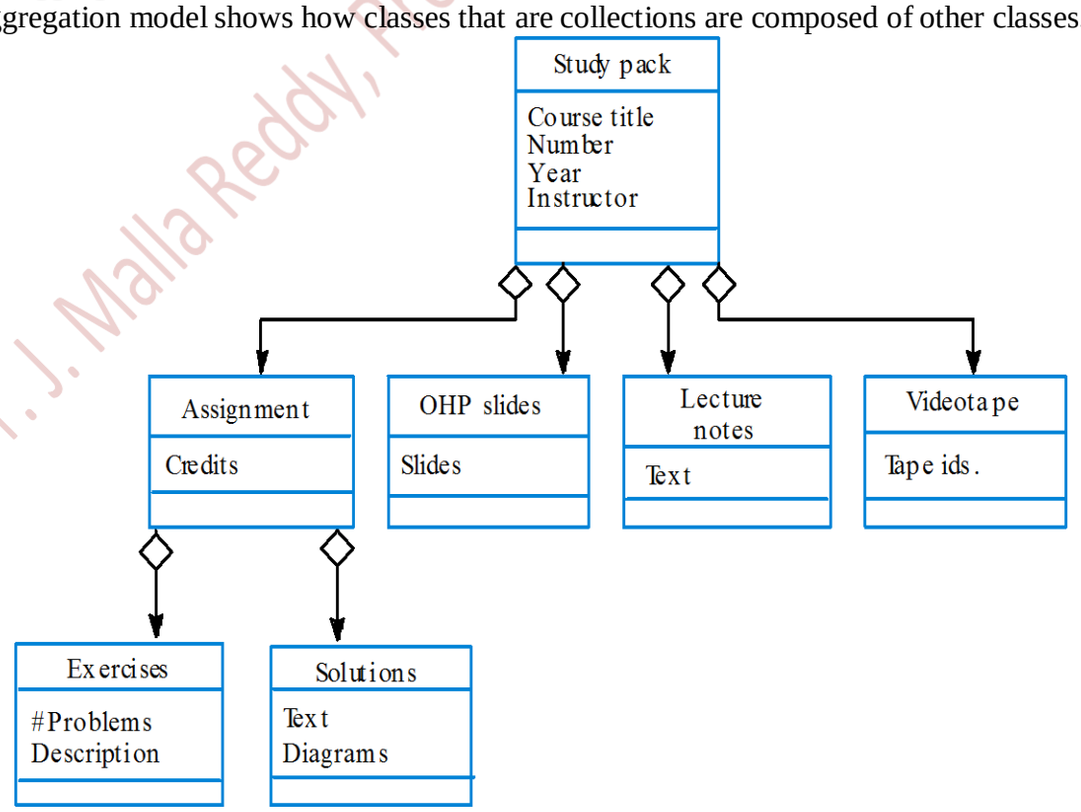
- This can lead to semantic conflicts where attributes/services with the same name in different super-classes have different semantics.

- Multiple inheritance makes class hierarchy reorganisation more complex.



Object aggregation :

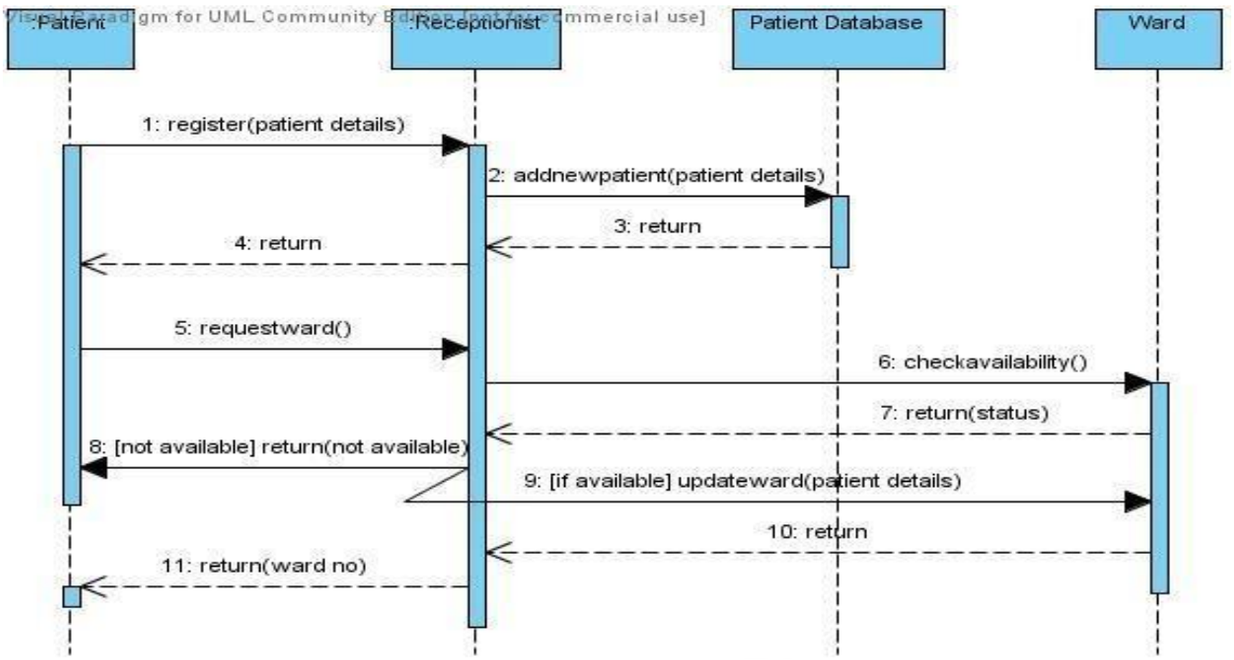
An aggregation model shows how classes that are collections are composed of other classes.



Aggregation models are similar to the part-of relationship in semantic data models

Object Behaviour modelling :

A behavioural model shows the interactions between objects to produce some particular system behaviour that is specified as a use-case.



Sequence/ Collaboration diagrams in the UML are used to model interaction between objects.

5. STRUCTURED METHODS

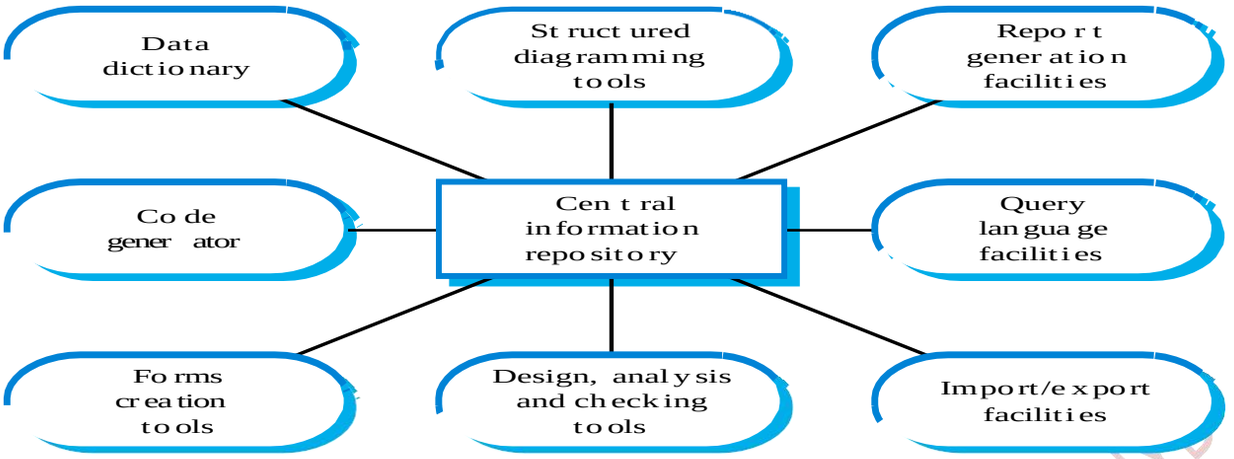
- Structured methods incorporate system modelling as an inherent part of the method.
- Methods define a set of models, a process for deriving these models and rules and guidelines that should apply to the models.
- CASE tools support system modelling as part of a structured method.

CASE workbenches

- A coherent set of tools that is designed to support related software process activities such as analysis, design or testing.
- Analysis and design workbenches support system modelling during both requirements engineering and system design.
- These workbenches may support a specific design method or may provide support for a creating several different types of system model.

Analysis workbench components :

- Diagram editors
- Model analysis and checking tools
- Repository and associated query language
- Data dictionary
- Report definition and generation tools
- Forms definition tools
- Import/export translators
- Code generation tools



An Analysis and Design Workbench

Overall Key points:

- A model is an abstract system view. Complementary types of model provide different system information.
- Context models show the position of a system in its environment with other systems and processes.
- Data flow models may be used to model the data processing in a system.
- State machine models model the system's behaviour in response to internal or external events.
- Semantic data models describe the logical structure of data which is imported to or exported by the systems.
- Object models describe logical system entities, their classification and aggregation.
- Sequence models show the interactions between actors and the system objects that they use.
- Structured methods provide a framework for developing system models.

IMPORTANT QUESTIONS

1. **Explain the importance of Requirement Engineering.**
2. **Distinguish between functional and Non functional Requirements.**
3. **Explain the various activities involved in requirement engineering process.**
4. **Explain about the various system models with diagrams.**