# FACULTY OF ENGINEERING

**B.E. 3/4 (CSE) I – Semester (Main)) Examination, November 2020**

**Subject: Software Engineering**

**1. Mention about software myths?**

**2. Define Project Plan?**

**3. Define Requirement engineering and Requirement plans?**

**4. What are Software process framework activities?**

**5.What are the risk management activities?**

**6.What are design Concepts?**

**7.What is Testing and write about Purpose of Testing?**

**8.Define User Interface Analysis?**

**9.What are characteristics of good Design?**

**10.What are perspective process models?**

**11.a. Explain CMMI Level?**
**11.B Explain Software Process?**

**12.Describe Requirement Engineering and Tasks?**

**13.A How are verification and validation important individually?**

**13.b. Describe Effort Estimation?**

**14. What are requirement engineering tasks? Explain validating requirements?**

**15. Discuss various analysis modelling approaches in detail.**

**16.Explain the concepts of Object-Oriented Analysis?**

**17. Write a short note on**

**A) Glass box Testing**

**B) Debugging**

**C) Fault-Based Testing**

**1.Mention about software myths?**

**Ans**. Software myths are misleading attitudes that have caused serious problems for managers and technical people alike.

There are three kinds of software myths:

**1)Management myths**: Managers with software responsibility are often under pressure to maintain budgets, keep schedules from slipping, and improve quality.

- **Myth:** We already have a book that's full of standards and procedures for building software, won't that provide my people with everything they need to know?
  **Reality:** The book of standards may very well exist, but isn't used. Most software practitioners aren't aware of its existence. Also, it doesn't reflect modern software engineering practices and is also complete.
  **Myth**: My people have state-of-the-art software development tools, after all, we buy them the newest computers.
  **Reality**: It takes much more than the latest model mainframe, workstation, or PC to do high-quality software development. Computer-aided software engineering (CASE) tools are more important than hardware for achieving good quality and productivity, yet the majority of software developers still do not use them effectively.

**2) Customer myths**: Customer myths lead to false expectations (by the customer) and ultimately, dissatisfaction with the developer. Following are the customer myths:

- **Myth**: A general statement of objectives is sufficient to begin writing programs-we can fill in the details later.
  **Reality:** A poor up-front definition is the major cause of failed software efforts. A formal and detailed description of the functions, behavior, performance, interfaces, design constraints, and validation criteria is essential.
  • **Myth**: Project requirements continually change, but change can be easily accommodated because software is flexible.
  **Reality**: It is true that software requirements change, but the impact of change varies with the time at which it is introduced. When changes are requested during software design, the cost impact grows rapidly. Resources have been committed and a design framework has been established. Change can cause heavy additional costs. Change, when requested after software is in production, can be much more expensive than the same change requested earlier.

**3) Practitioner's myths:** Practitioners have following myths:

- **Myth**: Once we write the program and get it to work, our job is done.
  **Reality**: Industry data indicate that between 60 and 80 percent of all effort expended on software will be expended after it is delivered to the customer for the first time.
  • **Myth:** Until I get the program "running" I have no way of assessing its quality.
  **Reality:** One of the most effective software quality assurance mechanisms can be applied from the inception of a project—the formal technical review.

**2. Define Project Plan?**

**Ans.** A project plan helps a project manager to understand, monitor, and control the development of software project. This plan is used as a means of communication between the users and project management team. There are various advantages associated with a project plan, some of which are listed below.

- It ensures that software is developed according to the user requirements, objectives, and scope of the project.
- It identifies the role of each project management team member involved in the project.
- It monitors the progress of the project according to the project plan.
- It determines the available resources and the activities to be performed during software development.
- It provides an overview to management about the costs of the software project, which are estimated during project planning.

There are differences in the contents of two project plans depending on the kind of project and user requirements.

1. **Introduction:** Describes the objectives of the project and provides information about the constraints that affect the software project.
2. **Project organization:** Describes the responsibilities assigned to the project management team members for completing the project.
3. **Risk analysis:** Describes the risks that can possibly arise during software development as well as explains how to assess and reduce the effect of risks.
4. **Resource requirements:** Specifies the hardware and software required to carry out the software project. Cost estimation is done according to these resource requirements.
5. **Work breakdown:** Describes the activities into which the project is divided. It also describes the milestones and deliverables of the project activities.
6. **Project schedule:** Specifies the dependencies of activities on each other. Based on this, the time required by the project management team members to complete the project activities is estimated.

**3. Define Requirement engineering and Requirement plans?**

**Ans.** Requirements engineering is the discipline concerned with establishing and managing requirements. It consists of requirements elicitation, analysis, specification, verification, and management. Keywords: analysis, definition, development, elicitation, management, requirements, systems engineering, verification.

**4. What are Software process framework activities?**

**Ans. Framework** is a Standard way to build and deploy applications. **Software Process Framework** is a foundation of complete software engineering process. Software process framework includes all set of umbrella activities. It also includes number of framework activities that are applicable to all software projects.

A generic process framework encompasses five activities which are given below one by one:

1. **Communication:**
   In this activity, heavy communication with customers and other stakeholders, requirement gathering is done.

2. **Planning:**
   In this activity, we discuss the technical related tasks, work schedule, risks, required resources etc.
3. **Modeling:**
   Modeling is about building representations of things in the 'real world'. In modeling activity, a product's model is created in order to better understanding and requirements.
4. **Construction:**
   In software engineering, construction is the application of set of procedures that are needed to assemble the product. In this activity, we generate the code and test the product in order to make better product.
5. **Deployment:**
   In this activity, a complete or non-complete products or software are represented to the customers to evaluate and give feedback. on the basis of their feedback we modify the products for supply better product.

**Umbrella activities include:**

- Risk management
- Software quality assurance (SQA)
- Software configuration management (SCM)
- Measurement
- Formal technical reviews (FTR)

**5.What are the risk management activities?**

Ans.

**Risk Identification:** The project organizer needs to anticipate the risk in the project as early as possible so that the impact of risk can be reduced by making effective risk management planning.

**Risk Analysis:** During the risk analysis process, you have to consider every identified risk and make a perception of the probability and seriousness of that risk.

**Risk prioritization**: The overall set of identified risk events, their impact assessments, and their probabilities of occurrences are "processed" to derive a most-to-least-critical rank-order of identified risks. A major purpose of **prioritizing risks** is to form a basis for allocating resources.

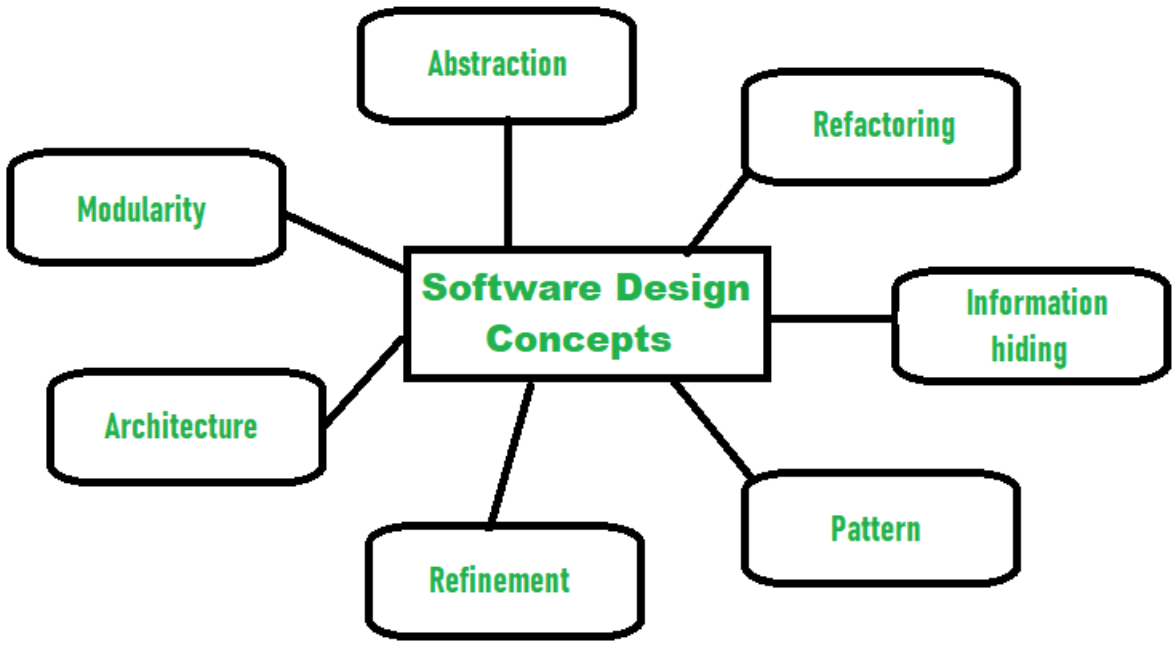
**Risk planning:** The risk planning method considers each of the key risks that have been identified and develop ways to maintain these risks.

**Risk Monitoring:** Risk monitoring is the method king that your assumption about the product, process, and business risks has not changed.

**Risk resolution**: It produces a situation in which the **risk** items are eliminated or otherwise resolved (for example **risk** avoidance via relaxation of requirements).

**6.What are design Concepts?**

**Ans.** The software design concept provides a supporting and essential structure or model for developing the right software. There are many concepts of software design and some of them are given below:




**7.What is Testing and write about Purpose of Testing?**

**Ans.** The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements. Some prefer saying Software testing definition as a White Box and Black Box Testing. In simple terms, Software Testing means the Verification of Application Under Test (AUT).

**8.Define User Interface Analysis?**

**Ans.** User interface (UI) design analysis is the study of how users use a particular product or system to achieve goals by performing tasks. User analysis studies users, who they are and what tasks they might need to perform. Use cases are primary examples of each action for which the user may use the system.

**9.What are characteristics of good Design?**

**Ans.** Characteristics of a Good Software Design

- Correctness: A good design should correctly implement all the functionalities identified in the SRS document.
- Understandability: A good design is easily understandable.
- Efficiency: It should be efficient.
- Maintainability: It should be easily amenable to change.

**10.What are perspective process models?**

**Ans.** A prescriptive process model is a model that describes "how to do" according to a certain software process system. A prescriptive model prescribes how a new software system should be developed.

**11.a. Explain CMMI Level?**

**Ans.** MMI's Maturity Levels are:

- **Maturity Level 0 – Incomplete:** At this stage work "may or may not get completed." Goals have not been established at this point and processes are only partly formed or do not meet the organizational needs.
- **Maturity Level 1 – Initial:** Processes are viewed as unpredictable and reactive. At this stage, "work gets completed but it's often delayed and over budget." This is the worst stage a business can find itself in — an unpredictable environment that increases risk and inefficiency.
- **Maturity Level 2 – Managed:** There's a level of project management achieved. Projects are "planned, performed, measured and controlled" at this level, but there are still a lot of issues to address.
- **Maturity Level 3 – Defined:** At this stage, organizations are more proactive than reactive. There's a set of "organization-wide standards" to "provide guidance across projects, programs and portfolios." Businesses understand their shortcomings, how to address them and what the goal is for improvement.
- **Maturity Level 4 – Quantitatively managed:** This stage is more measured and controlled. The organization is working off quantitative data to determine predictable processes that align with stakeholder needs. The business is ahead of risks, with more data-driven insight into process deficiencies.
- **Maturity Level 5 – Optimizing:** Here, an organization's processes are stable and flexible. At this final stage, an organization will be in constant state of improving and responding to changes or other opportunities. The organization is stable, which allows for more "agility and innovation," in a predictable environment.

Once organizations hit Levels 4 and 5, they are considered high maturity, where they are "continuously evolving, adapting and growing to meet the needs of stakeholders and customers." That is the goal of the CMMI: To create reliable environments, where products, services and departments are proactive, efficient and productive.

**CMMI Capability Levels**

The CMMI also has capability levels that are used to appraise an organization's performance and process improvement as it applies to an individual practice area outlined in the CMMI model. It can help bring structure to process and performance improvement and each level builds on the last, similar to the maturity levels for appraising an organization.

The capability levels are:

- **Capability Level 0 – Incomplete:** Inconsistent performance and an "incomplete approach to meeting the intent of the practice area."
- **Capability Level 1 – Initial:** The phase where organizations start to address performance issues in a specific practice area, but there is not a complete set of practices in place.
- **Capability Level 2 – Managed:** Progress is starting to show and there is a full set of practices in place that specifically address improvement in the practice area.
- **Capability Level 3 – Defined:** There's a focus on achieving project and organizational performance objectives and there are clear organizational standards in place for addressing projects in that practice area.

**11.B Explain Software Process?**

**Ans. Software** is the set of instructions in the form of programs to govern the computer system and to process the hardware components. To produce a software product the set of activities is used. This set is called a software process.

**Components of Software:**

There are three components of the software:

1. **Program:**
   A computer program is a list of instructions that tell a computer what to do.
2. **Documentation:**
   Source information about the product contained in design documents, detailed code comments, etc.
3. **Operating Procedures:**
   Set of step-by-step instructions compiled by an organization to help workers carry out complex routine operations.

There are four basic key process activities:

1. **Software Specifications:**
   In this process, detailed description of a software system to be developed with its functional and non-functional requirements.
2. **Software Development:**
   In this process, designing, programming, documenting, testing, and bug fixing is done.
3. **Software Validation:**
   In this process, evaluation software product is done to ensure that the software meets the business requirements as well as the end users needs.

4. **Software Evolution:**
   It is a process of developing software initially, then timely updating it for various reasons.

**Software Crisis:**

1. **Size and Cost:**
   Day to day growing complexity and expectation out of software. Software are more expensive and more complex.
2. **Quality:**
   Software products must have good quality.
3. **Delayed Delivery:**
   Software takes longer than the estimated time to develop, which in turn leads to cost shooting up.

**Software Process Model:**

A software process model is an abstraction of the actual process, which is being described. It can also be defined as a simplified representation of a software process. Each model represents a process from a specific perspective. Basic software process models on which different type of software process models can be implemented:

1. **A workflow Model:**
   It is the sequential series of tasks and decisions that make up a business process.
2. **The Waterfall Model:**
   It is a sequential design process in which progress is seen as flowing steadily downwards. Phases in waterfall model:
   - **(i)** Requirements Specification
   - **(ii)** Software Design
   - **(iii)** Implementation
   - **(iv)** Testing
3. **Dataflow Model:**
   It is diagrammatic representation of the flow and exchange of information within a system.
4. **Evolutionary Development Model:**
   Following activities are considered in this method:
   - **(i)** Specification
   - **(ii)** Development
   - **(iii)** Validation
5. **Role / Action Model:**
   Roles of the people involved in the software process and the activities.

**12.Describe Requirement Engineering and Tasks?**

**Ans.** Requirement Engineering is the process of defining, documenting and maintaining the requirements. It is a process of gathering and defining service provided by the system. Requirements Engineering Process consists of the following main activities:

- Requirements elicitation
- Requirements specification
- Requirements verification and validation
- Requirements management

**Requirements Elicitation:**

It is related to the various ways used to gain knowledge about the project domain and requirements. The various sources of domain knowledge include customers, business manuals, the existing software of same type, standards and other stakeholders of the

project.
The techniques used for requirements elicitation include interviews, brainstorming, task analysis, Delphi technique, prototyping, etc. Some of these are discussed Elicitation does not produce formal models of the requirements understood. Instead, it widens the domain knowledge of the analyst and thus helps in providing input to the next stage.

**Requirements specification:**

This activity is used to produce formal software requirement models. All the requirements including the functional as well as the non-functional requirements and the constraints are specified by these models in totality. During specification, more knowledge about the problem may be required which can again trigger the elicitation process.
The models used at this stage include ER diagrams, data flow diagrams (DFDs), function decomposition diagrams (FDDs), data dictionaries, etc.

**Requirements verification and validation:**

**Verification:** It refers to the set of tasks that ensures that the software correctly implements a specific function.

**Validation:** It refers to a different set of tasks that ensures that the software that has been built is traceable to customer requirements.
If requirements are not validated, errors in the requirement definitions would propagate to the successive stages resulting in a lot of modification and rework.
The main steps for this process include:

- The requirements should be consistent with all the other requirements i.e no two requirements should conflict with each other.
- The requirements should be complete in every sense.
- The requirements should be practically achievable.

**Requirements management:**

Requirement management is the process of analysing, documenting, tracking, prioritizing and agreeing on the requirement and controlling the communication to relevant stakeholders. This stage takes care of the changing nature of requirements. It should be ensured that the SRS is as modifiable as possible so as to incorporate changes in requirements specified by the end users at later stages too. Being able to modify the software as per requirements in a systematic and controlled manner is an extremely important part of the requirements engineering process.

**13.A How are verification and validation important individually?**

**Ans.** Validation addresses each individual requirement to ensure that it is:

- **Correct** – accurately states a customer or external need
- **Clear** – has only one possible meaning
- **Feasible** – can be implemented within known constraints
- **Modifiable** – can be easily changed, with history, when necessary
- **Necessary** – documents something customers really need
- **Prioritized** – ranked as to importance of inclusion in product
- **Traceable** – can be linked to system requirements, and to designs, code, and tests
- **Verifiable –** correct implementation can be determined by testing, inspection, analysis, or demonstration.

**Verification** is the process of confirming that the designed and built product fully addresses documented requirements. Verification consists of performing various inspections, tests, and analyses throughout the product lifecycle to ensure that the design, iterations, and the finished product fully addresses the requirements.

To prevent rework, requirements should be validated and approved before design. If the requirements are not validated, then requirement validation and product verification will inevitably be done together during product design and development activities. Because verification is guided by requirements, there is high likelihood that missing or invalid requirements will not be discovered. Missing and invalid requirements causes rework and significant cost overruns.

Note that the last quality characteristic listed above is "Verifiable." Assessing verification as you develop your requirements and validating that the requirements are verifiable significantly improves requirement quality.

**13.b. Describe Effort Estimation?**

**Ans. Effort estimation** is the process of predicting the most realistic amount of effort (expressed in terms of person-hours or money) required to develop or maintain software based on incomplete, uncertain and noisy input. Effort estimates may be used as input to project plans, iteration plans, budgets, investment analyses, pricing processes and bidding rounds.

- Expert estimation: The quantification step, i.e., the step where the estimate is produced based on judgmental processes.
- Formal estimation model: The quantification step is based on mechanical processes, e.g., the use of a formula derived from historical data.
- Combination-based estimation: The quantification step is based on a judgmental and mechanical combination of estimates from different sources.

**Accuracy**

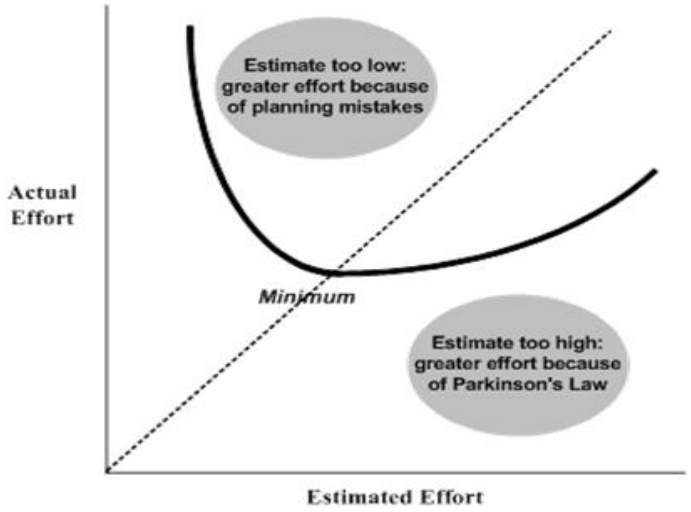
Effort estimation accuracy depends on available information. Usually, you have less information before you start the project (presales) and you have more information while working in the project.

Most of the times, you can have more accurate effort estimation after requirement analysis. However, initial effort estimation at early project stages is sometimes more important. E.g. you give a financial offer based on early stage effort estimation. The price you will give will probably bind you for the whole project, so it is important to have a good estimation from the beginning.

Although it is obvious that accurate effort estimation is crucial, most of the times people fail to predict well. Actually, it is amazing how often and how much effort estimation goes wrong.

Approximately 40% of industry software projects that get cancelled are cancelled due to, partly or completely, failures in effort estimation.

- It is a common fact that the larger the project is, the more essential is to have a good estimation and, at the same time, the more difficult is to have one.
- Have in mind that both low effort estimation and high effort estimation cause troubles, and make the project take longer to complete.




**14. What are requirement engineering tasks? Explain validating requirements?**

**Ans.** The process of collecting the software requirement from the client then understand, evaluate and document it is called as requirement engineering.

Requirement engineering constructs a bridge for design and construction.

**Requirement engineering consists of seven different tasks as follow:**

**1. Inception**

- Inception is a task where the requirement engineering asks a set of questions to establish a software process.
- In this task, it understands the problem and evaluates with the proper solution.
- It collaborates with the relationship between the customer and the developer.
- The developer and customer decide the overall scope and the nature of the question.

**2. Elicitation**

Elicitation means to find the requirements from anybody.

The requirements are difficult because the **following problems occur in elicitation**.

**Problem of scope:** The customer give the unnecessary technical detail rather than clarity of the overall system objective.

**Problem of understanding:** Poor understanding between the customer and the developer regarding various aspect of the project like capability, limitation of the computing

environment.

**Problem of volatility:** In this problem, the requirements change from time to time and it is difficult while developing the project.

**3. Elaboration**

- In this task, the information taken from user during inception and elaboration and are expanded and refined in elaboration.
- Its main task is developing pure model of software using functions, feature and constraints of a software.

**4. Negotiation**

- In negotiation task, a software engineer decides the how will the project be achieved with limited business resources.
- To create rough guesses of development and access the impact of the requirement on the project cost and delivery time.

**5. Specification**

- In this task, the requirement engineer constructs a final work product.
- The work product is in the form of software requirement specification.
- In this task, formalize the requirement of the proposed software such as informative, functional and behavioural.
- The requirement is formalized in both graphical and textual formats.

**6. Validation**

- The work product is built as an output of the requirement engineering and that is accessed for the quality through a validation step.
- The formal technical reviews from the software engineer, customer and other stakeholders helps for the primary requirements validation mechanism.

**7. Requirement management**

- It is a set of activities that help the project team to identify, control and track the requirements and changes can be made to the requirements at any time of the ongoing project.
- These tasks start with the identification and assign a unique identifier to each of the requirement.
- After finalizing the requirement traceability table is developed.
- The examples of traceability table are the features, sources, dependencies, subsystems and interface of the requirement.

checking that requirements defined for development, define the system that the customer really wants. To check issues related to requirements, we perform requirements validation.

In the requirements validation process, we perform a different type of test to check the requirements mentioned in the Software Requirements Specification (SRS), these checks include:

- Completeness checks
- Consistency checks
- Validity checks
- Realism checks
- Ambiguity checks
- Verifiability

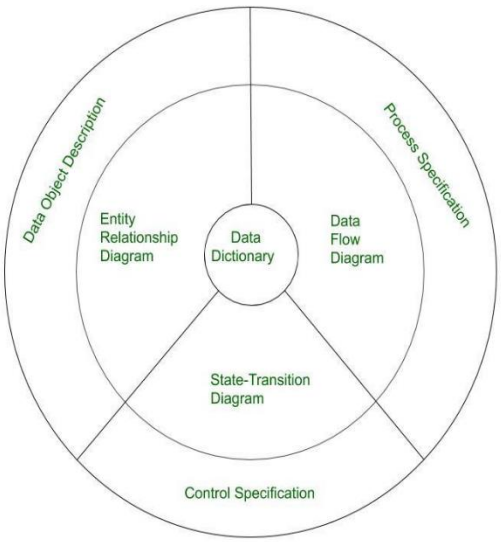
**15. Discuss various analysis modelling approaches in detail.**

**Ans. Analysis Model** is a technical representation of the system. It acts as a link between system description and design model. In Analysis Modelling, information, behavior and functions of the system is defined and translated into the architecture, component and interface level design in the design modeling.

**Objectives of Analysis Modelling:**

1. It must establish a way of creation of software design.
2. It must describe requirements of customer.
3. It must define set of requirements which can be validated, once the software is built.

**Elements of Analysis Model:**




**1. Data Dictionary:**

It is a repository that consists of description of all data objects used or produced by software. It stores the collection of data present in the software. It is a very crucial element of the analysis model. It acts as a centralized repository and also helps in modelling of data objects defined during software requirements.

1. **Entity Relationship Diagram (ERD):**
   It depicts relationship between data objects and used in conducting of data modelling activity. The attributes of each object in the Entity Relationship Diagram can be described using Data object description. It provides the basis for activity related to data design.

2. **Data Flow Diagram (DFD):**
   It depicts the functions that transform data flow and it also shows how data is transformed when moving from input to output. It provides the additional information which is used during the analysis of information domain and serves as a basis for the modelling of function. It also enables the engineer to develop models of functional and information domain at the same time.

3. **State Transition Diagram:**
   It shows various modes of behaviour (states) of the system and also shows the transitions from one state to other state in the system. It also provides the details of how system behaves due to the consequences of external events. It represents the behaviour of a system by presenting its states and the events that cause the system to change state. It also describes what actions are taken due to the occurrence of a particular event.

4. **Process Specification:**
   It stores the description of each functions present in the data flow diagram. It describes the input to a function, the algorithm that is applied for transformation of input, and the output that is produced. It also shows regulations and barriers imposed on the performance characteristics that are applicable to the process, and layout constraints that could influence the way in which the process will be implemented.

5. **Control Specification:**
   It stores the additional information about the control aspects of the software. It is used to indicate how the software behaves when an event occurs and which processes are invoked due to the occurrence of the event. It also provides the details of the processes which are executed to manage events.

6. **Data Object Description:**
   It stores and provides the complete knowledge about a data object present and used in the software. It also gives us the details of attributes of the data object present in Entity Relationship Diagram. Hence, it incorporates all the data objects and its attributes.

**16.Explain the concepts of Object-Oriented Analysis?**

**Ans.** The intent of object-oriented analysis (OOA)is to define all classes(relationships & behavior associated with them) that are relevant to the problem to be solved.

- To accomplish this, no. of tasks must occur

1. Basic user requirements must be communicated b/w customer and s/w engineer.

2. Classes must be identified(i.e. attributes and methods are defined).
3. A class hierarchy is defined.
4. Object Oriented relationships(object-connections) should be represented.
5. Object behaviour must be modelled.
6. TASK 1 through TASK 5 are reapplied iteratively until the model is complete.
7. Use-cases] are simply an aid to defining what exists outside the system (actors) and what should be performed by the system (use-cases)." Ivar Jacobson.

**Writing use-cases**

(1) What should we write about?

(2) How much should we write about it?

(3) How detailed should we make our description?

(4) How should we organize the description?

**Use-Cases:**

- a scenario that describes a "thread of usage" for a system
- actors represent roles people or devices play as the system functions
- users can play a number of different roles for a given scenario

**17. Write a short note on**

**A) Glass box Testing**

**B) Debugging**

**C) Fault-Based Testing**

**Ans.** Glass box testing is a testing technique that examines the program structure and derives test data from the program logic/code. The other names of glass box testing are clear box testing, open box testing, logic driven testing or path driven testing or structural testing.

Glass Box Testing Techniques:

- **Statement Coverage -** This technique is aimed at exercising all programming statements with minimal tests.
- **Branch Coverage -** This technique is running a series of tests to ensure that all branches are tested at least once.
- **Path Coverage -** This technique corresponds to testing all possible paths which means that each statement and branch is covered.

- Statement Testing = (Number of Statements Exercised / Total Number of Statements) x 100 %
- 
- Branch Testing = (Number of decisions outcomes tested / Total Number of decision Outcomes) x 100 %
- 
- Path Coverage = (Number paths exercised / Total Number of paths in the program) x 100 %

**Advantages of Glass Box Testing**:

- Forces test developer to reason carefully about implementation.
- Reveals errors in "hidden" code.
- Spots the Dead Code or other issues with respect to best programming practices.

**Disadvantages of Glass Box Testing:**

- Expensive as one has to spend both time and money to perform white box testing.
- Every possibility that few lines of code is missed accidentally.
- In-depth knowledge about the programming language is necessary to perform white box testing.

**B) Debugging**

Debugging is the process of detecting and removing of existing and potential errors (also called as ‘bugs’) in a software code that can cause it to behave unexpectedly or crash. To prevent incorrect operation of a software or system, debugging is used to find and resolve bugs or defects. When various subsystems or modules are tightly coupled, debugging becomes harder as any change in one module may cause more bugs to appear in another. Sometimes it takes more time to debug a program than to code it.

**C) Fault-Based Testing**

A model of potential program faults is a valuable source of information for evaluating and designing test suites. Some fault knowledge is commonly used in functional and structural testing, for example when identifying singleton and error values for parameter characteristics in category partition testing, or when populating catalogs with erroneous values, but a fault model can also be used more directly. Fault-based testing uses a fault model directly to hypothesize potential faults in a program under test, and to create or evaluate test suites based on its efficacy in detecting those hypothetical faults