IV Sem BCA: Software Engineering

STUDY MATERIAL IN SIMPLE STEPS

Faculty Name: P YOGANANDA **Depot: C.S. & Applications.**

Chapter-1 **Introduction to Software Engineering**

1) **Define Software Engineering?**

Software engineering is defined as a process of analyzing user requirements and then designing, building, and testing.

The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software; that is, the application of engineering to software.

2) **Define Software?**

Software, instructions that tell a computer what to do. **Software** comprises the entire set of programs, procedures, and routines associated with the operation of a computer system. ... A set of instructions that directs a computer's hardware to perform a task is called a program, or **software** program.

3) **Define System Software?**

System software is used for operating computer hardware. On other hand Application software is used by user to perform specific task. It provides platform for running application software. On other hand in application software can't runindependently.

4) **Define Software Development Life Cycle?**

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

5) **Define Software Process?**

A software process (also knows as software methodology) is a set of related activities that leads to the production of the software. These activities may involve the development of the software from the scratch, or, modifying an existing system.

6) Explain the components of Software Engineering?

- Functionality
- Reliability.
- Efficiency.
- Usability.
- Maintainability.
- Portability.

7) Explain the phases of SDLC?

It's typically divided into six to eight **steps**: Planning, Requirements, Design, Build, Document, Test, Deploy, Maintain. Some project managers will combine, split, or omit **steps**, depending on the project's scope. These are the core components recommended for all **software development** projects.

8) Define customized Product?

Customer software development is done to develop a software product as per the needs of particular customer.
In this development process, the the end-user requirements can be aggregated by communicating by them.

9) Explain Waterfall Model?

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, theoutcome of one phase acts as the input for the next phase sequentially.

The sequential phases in Waterfall model are −

- **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

- **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

10) Explain Iterative model?

Iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.

Iterative and Incremental development is a combination of both iterative design oriterative method and incremental build model for development. "During software development, more than one iteration of the software development cycle may be in progress at the same time." This process may be described as an "evolutionary acquisition" or "incremental build" approach."

In this incremental model, the whole requirement is divided into various builds. During each iteration, the development module goes through the requirements, design, implementation and testing phases. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is ready as per the requirement.

The key to a successful use of an iterative software development lifecycle is rigorous validation of requirements, and verification & testing of each version of the software against those requirements within each cycle of the model. As the software evolves through successive cycles, tests must be repeated and extended to verify each version of the software.

11) Explain Prototype model?

Prototype is a working model of software with some limited functionality. The prototype does not always hold the exact logic used in the actual software application and is an extra effort to be considered under effort estimation.

Following is a stepwise approach explained to design a software prototype.

- **Basic Requirement Identification**

This step involves understanding the very basics product requirements especially in terms of user interface. The more intricate details of the internal design and external aspects like performance and security can be ignored at this stage.

- **Developing the initial Prototype**

The initial Prototype is developed in this stage, where the very basic requirements are showcased and user interfaces are provided. These features may not exactly work in the same manner internally in the actual software developed. While, the workarounds are used to give the same look and feel to the customer in the prototype developed.

- **Review of the Prototype**

The prototype developed is then presented to the customer and the other important

stakeholders in the project. The feedback is collected in an organized manner and used for further enhancements in the product under development.

- **Revise and Enhance the Prototype**

The feedback and the review comments are discussed during this stage and some negotiations happen with the customer based on factors like – time and budget constraints and technical feasibility of the actual implementation. The changes accepted are again incorporated in the new Prototype developed and the cycle repeats until the customer expectations are met.

Chapter-2 & 3 System Engineering and Software Requirement Specification Document.

1) **Define system Engineering?**
   Systems engineering is an interdisciplinary field of engineering and engineering management that focuses on how to design, integrate, and manage complex systems over their life cycles. At its core, systems engineering utilizes systems thinking principles to organize this body of knowledge. The individual outcome of such efforts, an engineered system, can be defined as a combination of components that work in synergy to collectively perform a useful function.

2) **Define emergent properties?**

   Emergent properties are properties that are characteristic of the system as a whole and not its component parts. The systems engineering process includes specification, design, development, integration and testing.

3) **What is system integration?**

   System integration involves integrating existing, often disparate systems in such a way "that focuses on increasing value to the customer" (e.g., improved product quality and performance) while at the same time providing value to the company (e.g., reducing operational costs and improving response time).

4) **What is system Procurement?**

   A procurement system or purchasing system allows organizations to automate the process of purchasing goods/services and maintaining inventory.It helps manage all the procurement-related processes, including: Generating purchase orders. Selecting and managing vendors. Approving delivered goods/services.

5) **What are the problems during installation of software?**

6) **What are the phases of system engineering process?**

Incorrect calculations - This is seen in functions such as financial and datecalculations. The key determinant is whenever mathematical functions and mathematical operators are involved.

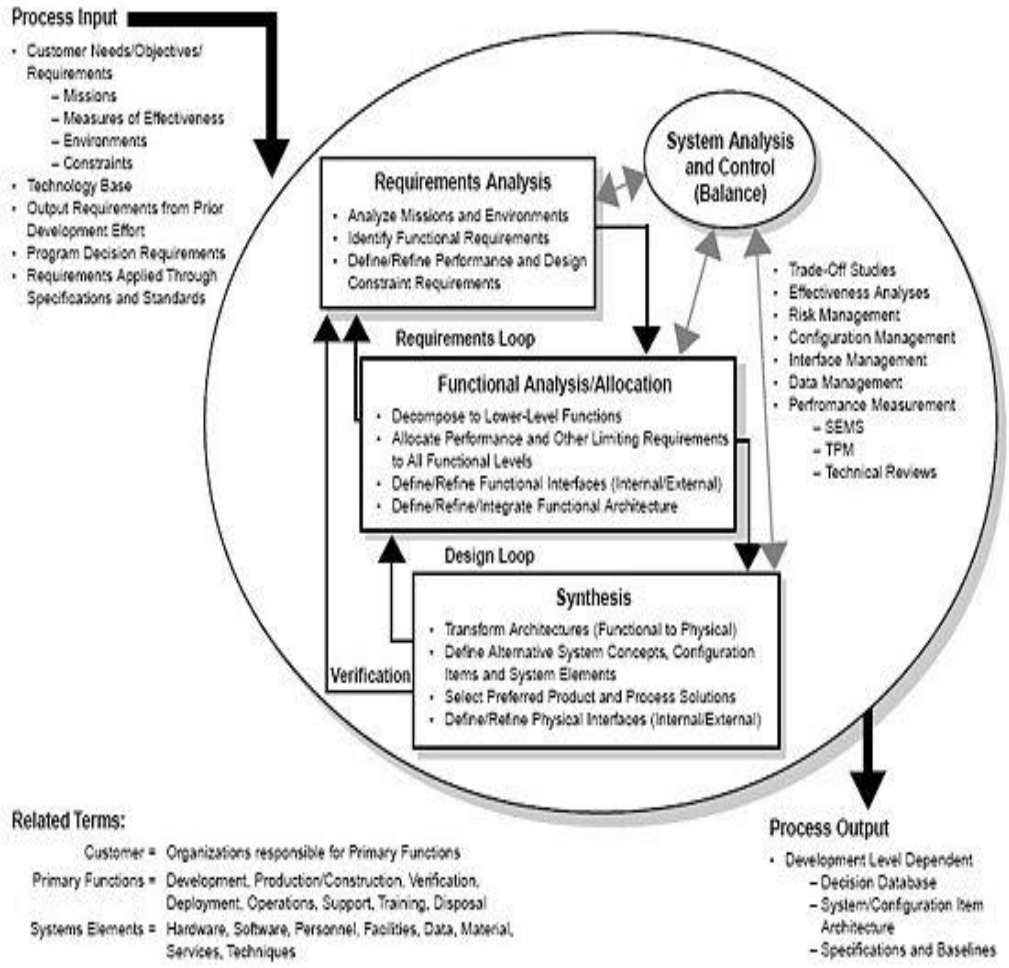
Incorrect data edits - This is when the software does not apply existing dataedits correctly

Ineffective data edits - This is when data edits are in place and working correctly, yet still fail to prevent

The Systems Engineering Process is a comprehensive, iterative and recursive problem solving process, applied sequentially top-down by integrated teams. It transforms needs and requirements into a set of system product and process descriptions, generate information for decision makers, and provides input for the next level of development. The process is applied sequentially, one level at a time, adding additional detail and definition with each level of development.

The four (4) steps that comprise the SE Process are:

Step 1: Requirements Analysis
Step 2: System Analysis Control
Step 3: Functional Analysis/Allocation
Step 4: Design Synthesis



Chapter-3 Software Requirement Specification Document

1) What is requirement?

The software requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client's point of view.

2) Define Software requirement specification?

A software requirements specification (SRS) is a document that captures complete description about how the system is expected to perform. It isusually signed off at the end of requirements engineering phase.

3) What are functional requirements?

In software engineering and systems engineering, a functional requirement defines a function of a system or its component, where a function is described as a specification of behavior between outputs and inputs..... As defined in requirements engineering, functional requirements specify particular results of a system.

4) What are non-functional requirements?

These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are alsocalled non-behavioral requirements.

5) What are the components of SRS?

Functional Requirements. Functional requirements specify what **output** should be produced from the given inputs. Performance Requirements (**Speed** Requirements) This part of an SRS specifies the performance constraints on the software system. Design Constraints. External Interface Requirements.

6) Explain IEEE Structure of SRS?

In order to form a good SRS, here you will see some points which can be used and should be considered to form a structure of good SRS. These are as follows :
Introduction

(i) Purpose of this document
(ii) Scope of this document
(iii) Overview

- General description
  Functional Requirements
  Interface Requirements

Performance Requirements
Design Constraints
Non-Functional Attributes
Preliminary Schedule and Budget
Appendices

Depending upon information gathered after interaction, SRS is developed which describes requirements of software that may include changes and modifications that is needed to be done to increase quality of product and to satisfy customer's demand.

Introduction :

- **(i) Purpose of this Document –**
  At first, main aim of why this document is necessary and what's purpose of document is explained and described.

- **(ii) Scope of this document –**
  In this, overall working and main objective of document and what value it will provide to customer is described and explained. It also includes a description of development cost and time required.

- **(iii)Overview –**
  In this, description of product is explained. It's simply summary or overall review of product.

- **General description :**
  In this, general functions of product which includes objective of user, a user characteristic, features, benefits, about why its importance is mentioned. It also describes features of user community.

- **Functional Requirements:**
  In this, possible outcome of software system which includes effects due to operation of program is fully explained. All functional requirements which may include calculations, data processing, etc. are placed in a ranked order.
- **Interface Requirements:**
  In this, software interfaces which mean how software program communicates with each other or users either in form of any language, code, or message are fully described and explained. Examples can be shared memory, data streams, etc.

- **Performance Requirements :**
  In this, how a software system performs desired functions under specific condition is explained. It also explains required time, required memory, maximum error rate, etc.

- **Design Constraints :**
  In this, constraints which simply means limitation or restriction are specified and explained for design team. Examples may include use of a particular algorithm, hardware and software limitations, etc.

- **Non-Functional Attributes :**
  In this, non-functional attributes are explained that are required by software system for better performance. An example may include Security, Portability, Reliability, Reusability, Application compatibility, Data integrity, Scalability capacity, etc.

- **Preliminary Schedule and Budget :**
  In this, initial version and budget of project plan are explained which include overall time duration required and overall cost required for development of project.

- **Appendices :**
  In this, additional information like references from where information is gathered, definitions of some specific terms, acronyms, abbreviations, etc. are given and explained.

7) Explain Requirement engineering process?

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 consistsof the following main activities:

Requirements elicitation
Requirements specification
Requirements verification and validation
Requirements management

9) Explain Requirement Management?

- The purpose of **requirements management** is to ensure product development goals are successfully met. It is a set of techniques for documenting, analyzing, prioritizing, and

agreeing on **requirements** so that **engineering** teams always have current and approved **requirements**.

- It includes various activities:
- Planning requirement phase
- Establishing the requirements process
- Controlling requirement changes
- Minimizing the addition of new requirements
- Tracking progress
- Resolving issues
- Holding requirement reviews
- The following are the key requirement skills:
- Analyze the problem
- Understand stakeholders
- Define the system
- Manage the scope of the system
- Refine the system definition
- Manage changing requirements

10) Define context model, behavior model and semantic model?

context Model:

A context model (or context modeling) defines how context data are structured and maintained (It plays a key role in supporting efficient context management). key role of context model is to simplify and introduce greater structure into the task of developing context-aware applications.

Behavior Model:

A behavioural model shows the interactions between objects to produce some particular system behaviour that is specified as a use-case. Sequence diagrams (or collaboration diagrams) in the UML are used to model interaction between objects.

Semantic Model:

A semantic data model in software engineering has various meanings: It is a conceptual data model in which semantic information is included. This means that the model describes the meaning of its instances. Such semantic models are fact-oriented.

11) Explain the types of object model?

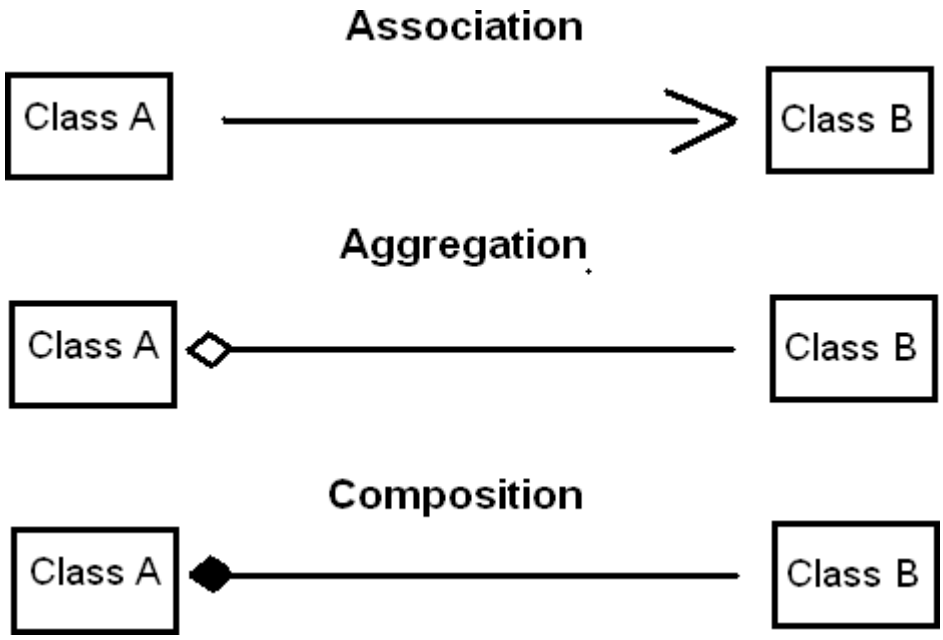
There are 3 types of object model

1) Inheritance model:

- Inheritance in the object model is a means of defining one class in terms of another. This is common usage for most of us. For example, a conifer is a type of tree. There are certain characteristics that are true for all trees, yet there are specific characteristics for conifers.

2) Object Aggregation:

An aggregation is a subtype of an association relationship in UML. Aggregation and composition are both the types of association relationship in UML. An aggregation relationship can be described in simple words as "an object of one class can own or access the objects of another class.



3) Object Behavior Modeling:

Modelling Behaviours • A behavioural model shows the interactions between objects to produce some particular system behaviour that is specified as a use-case. Sequence diagrams, Activity diagrams, state diagrams, collaboration diagrams are usedto model interaction between objects.

Behavior of a system is explained and represented with the help of a diagram. This diagram is known as State Transition Diagram. It is a collection of states and events. It

usually describes overall states that a system can have and events which are responsible for a change in state of a system.

So, on some occurrence of a particular event, an action is taken and what action needs to be taken is represented by State Transition Diagram.

Chapter 4 Software Prototyping

1) **Define Prototype?**

In software development, a prototype is a rudimentary working model of a product or information system, usually built for demonstration purposes or as part of the development process. In the systems development life cycle (SDLC) Prototyping Model, a basic version of the system is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can now be developed.

2) **What are the characteristics of prototype?**

The basic idea in Prototype model is that instead of freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements.

3) **What is a prototype process?**

Prototyping process is defined as the process of developing a working replication of a product or system that has to be engineered. It offers a small scale facsimile of the end product and is used for obtaining customer feedback

4) **Define Evolutionary Prototyping?**

Evolutionary model is a combination of Iterative and Incremental model of software development life cycle. Delivering your system in a big bang release, delivering it in incremental process over time is the action done in this model. Some initial requirements and architecture envisioning need to be done.

5) **Define throw-away prototyping?**

It will not form part of the final solution Its likely to inform the final solution, but the prototype itself will not become part of the final solution.

6) **What are the objectives of prototyping?**

Objective of evolutionary prototype:

To deliver a working system to end users. The development starts with thoserequirements which are the best.

Objective of evolutionary prototype:

To validate or derive the system requirements. The prototyping process starts with those requirements which are poorly understood.

7) **What are the advantages of prototype model ?**

- This model is flexible in design.
- It is easy to detect errors.
- We can find missing functionality easily.
- There is scope of refinement, it means new requirements can be easily accommodated.
- It can be reused by the developer for more complicated projects in the future.
- It ensures a greater level of customer satisfaction and comfort.
- It is ideal for online system.
- It helps developers and users both understand the system better.
- Integration requirements are very well understood and deployment channels are decided at a very early stage.
- It can actively involve users in the development phase.

8) What is software design?

Software design is a process to transform user requirements into some suitable form, which helps the programmer in software coding and implementation. .Software design is the first step in SDLC (Software Design Life Cycle), which moves the concentration from problem domain to solution domain.

9) What is software design process?

It describes a system that will be able to accomplish the established tasks in the requirement analysis phase.

Design is defined as both the "process of defining the architecture, components, interfaces and other characteristics of a system or component and the result for [that] process."

10) Why is software design process important?

Software Design gives a foundation to construct your software structure. Hence, it is an essential step. It precedes creating or enforcing the product in every engineering field.

Design is the initial and the most crucial phase of the software development methodology, as the entire application will build upon the choices made in this phase.

11) Define Cohesion and coupling?

cohesion refers to the degree to which the elements inside a module belong together. In one sense, it is a measure of the strength of relationship between the methods and data of a class and some unifying purpose or concept served by that class.

coupling is the degree of interdependence between **software** modules; a measure of how closely connected two routines or modules are; the strength of the relationships between modules. Low **coupling** often correlates with high cohesion, and vice versa.

12) What are the two types of coupling?

Tight Coupling is the idea of binding resources to specific purposes and functions. Tightly-coupled components may bind each resource to a particular use case, a specific interface, or a specific frontend.

Loose coupling: In a loosely coupled system, the components are detached from each

other. Every resource could have multiple frontends or applications. The inverse is true of each of those elements as well. All systems can work independently, as part of the larger group of systems, or in close concert with multiple segmented groups of systems.

13) What are the principles of good design?

The principles of a good software design are:

Modularity:

Dividing a large software project into smaller portions/modules is known as modularity.

The project is divided into various components and work on one component is done at once. It becomes easy to test each component due to modularity.

Coupling:

Coupling refers to the extent of interdependence between software modules and how closely two modules are connected. Low coupling is a feature of good design. With low coupling, changes can be made in each module individually, without changing the other modules.

Abstraction:

The process of identifying the essential behavior by separating it from its implementation and removing irrelevant details is known as Abstraction. The inability to separate essential behavior from its implementation will lead to unnecessary coupling.

Anticipation of change:

The demands of software keep on changing, resulting in continuous changes in requirements as well.

Simplicity:

The aim of good software design is simplicity. Each task has its own module, which can be utilized and modified independently. It makes the code easy to use and minimizes the number of setbacks.

Sufficiency and completeness:

A good software design ensures the sufficiency and completeness of the software concerning the established requirements. It makes sure that the software has been adequately and wholly built.

14) Explain Design Priciples?

A particular area provided by design principle for the judgments of particular aspects ofdesign. We have three types of principles which are explained below:

1. **Division of problems** - The base of these principles is to divide a big problem in to little parts. Every little part developed by different programs individually. Every little part can be individually altered.
    - This helps the system to become more sufficient.
    - This principle reduce the size of the problem and make simple and easy to service or maintenance.
    - Leads to hierarchy in the design.

    For the solution of big problem it is necessary to became proper coordination between these small pieces of problems.
2. **Abstraction** - To get the information in concerned to software parts from the outside is called the abstraction.
3. **Top down and bottom up design planning** - According to this principle a big problem divided in two little parts which is called modules and solved these modules one by one individually so that no one module can effected each other. We have two types of approaches. The top down approach goes from high level to the lower level. On the

other side the bottom up approach goes the opposite that mean it goes lower level to top level.

Modularity- Dividing a large software project into smaller portions/modules is known as modularity.

The project is divided into various components and work on one component is done at once. It becomes easy to test each component due to modularity.

15) Explain various levels of cohesion ?

Cohesion is a measure that defines the degree of intra-dependability within elements of a module. The greater the cohesion, the better is the program design.

- **Co-incidental cohesion -** It is unplanned and random cohesion, which might be the result of breaking the program into smaller modules for the sake of modularization. Because it is unplanned, it may serve confusion to the programmers and is generally not-accepted.
- **Logical cohesion -** When logically categorized elements are put together into a module, it is called logical cohesion.
- **Temporal Cohesion -** When elements of module are organized such that they are processed at a similar point in time, it is called temporal cohesion.
- **Procedural cohesion -** When elements of module are grouped together, which are executed sequentially in order to perform a task, it is called procedural cohesion.
- **Communicational cohesion -** When elements of module are grouped together, which are executed sequentially and work on same data (information), it is called communicational cohesion.
- **Sequential cohesion -** When elements of module are grouped because the output of one element serves as input to another and so on, it is called sequential cohesion.
- **Functional cohesion -** It is considered to be the highest degree of cohesion, and it is highly expected. Elements of module in functional cohesion are grouped because they all contribute to a single well-defined function. It can also be reused.

Chapter 6 – Object Oriented Design & Function Oriented Design

1) What is object oriented design?

Object–oriented design can be identified as −

- Definition of the context of the system
- Designing system architecture
- Identification of the objects in the system
- Construction of design models
- Specification of object interfaces

2) What is object oriented analysis?

OOA introduces new concepts to investigate a problem. It is based in a set of basic principles, which are as follows-

- The information domain is modeled.
- Behavior is represented.
- Function is described.
- Data, functional, and behavioral models
- Early models represent the essence of the problem,

3) What is object class?

In object-oriented programming, a class is an extensible program-code-template for creating objects, providing initial values for state (member variables) and implementations of behavior (member functions or methods).

4) Describe object association model?

Association is a semantically weak relationship (a semantic dependency) between otherwise unrelated objects.

An association is a "using" relationship between two or more objects in which the objects have their own lifetime and there is no owner.

As an example, imagine the relationship between a doctor and a patient. A doctor can be associated with multiple patients.

At the same time, one patient can visit multiple doctors for treatment or consultation.

Each of these objects has its own life cycle and there is no "owner" or parent. The objects that are part of the association relationship can be created and destroyed independently.

In UML an association relationship is represented by a single arrow.

An association relationship can be represented as one-to-one, one-to-many, or many-to-many (also known as cardinality).

Essentially, an association relationship between two or more objects denotes a path of communication (also called a link) between them so that one object can send a message to another.

5) Define inheritance?

In object-oriented programming, inheritance is the mechanism of basing an object or class upon another object (prototype-based inheritance) or class (class- based inheritance), retaining similar implementation. ... An inherited class is called a subclass of its parent class or super class.

6) Describe object identification?

Object identification in **object** models means that every **object** instance has a unique, unchanging identity.

Object identification is often referred to as an OID. OIDs are used to reference **object** instances. Characteristics of OIDs: OIDs are independent of data contained in the **object**.

7) What is function oriented design?

Function Oriented Design is an approach to software design where the design is decomposed into a set of interacting units where each unit has a clearly defined function.

Generic Procedure:

Start with a high level description of what the software / program does. Refine each part of the description one by one by specifying in greater details the functionality of each part. These points lead to Top-Down Structure.

Function Oriented Design Strategies:

Function Oriented Design Strategies are as follows:

1. Data Flow Diagram (DFD)**:**

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.

2. Data Dictionaries:

Data dictionaries are simply repositories to store information about all data items defined in DFDs. At the requirement stage, data dictionaries contains data items. Data dictionaries include Name of the item, Aliases (Other names for items), Description / purpose, Related data items, Range of values, Data structure definition / form.

3. Structure Charts:

It is the hierarchical representation of system which partitions the system into black boxes (functionality is known to users but inner details are unknown). Components are read from top to bottom and left to right. When a module calls another, it views the called module as black box, passing required parameters and receiving results.

Pseudo Code:

Pseudo Code is system description in short English like phrases describing the function. It

use keyword and indentation. Pseudo codes are used as replacement for flow charts. It decreases the amount of documentation required.

8) Explain Structural decomposition in OOD?

" Divide and Conquer " is a commonly used technique for attacking design problems. It is particularly useful when modeling. If an object can be structurally decomposed into logical parts, then creating each of those parts becomes a somewhat simpler design problem. Constructing the complex object now requires solving each of these simpler problems. Additionally, modification of the objects becomes much easier as the code becomes more modular.

Various steps involved in structural decomposition:

Identify system processing transformations

Identify input transformations

Identify output transformations

Design composition environment along dimensions:

- Concreteness, which defines how the design component evolves from abstraction to concreteness.
- Specificity, which defines how the design component evolves from being generic to domain-specific.

9) What is object oriented design process?

Structured design processes involve Structured design processes involve developing a number of different system models.

They require a lot of effort for development and maintenance of these models and, for small systems this may not be cost small systems, this may not be cost effective.Highlights key activities without being tied to Highlights key activities without beingtied to any proprietary process such as the RUP.

• Define the context and modes of use of the Define the context and modes of use of the system;

• Design the system architecture;

• Identify the principal system objects;

• Develop design models;

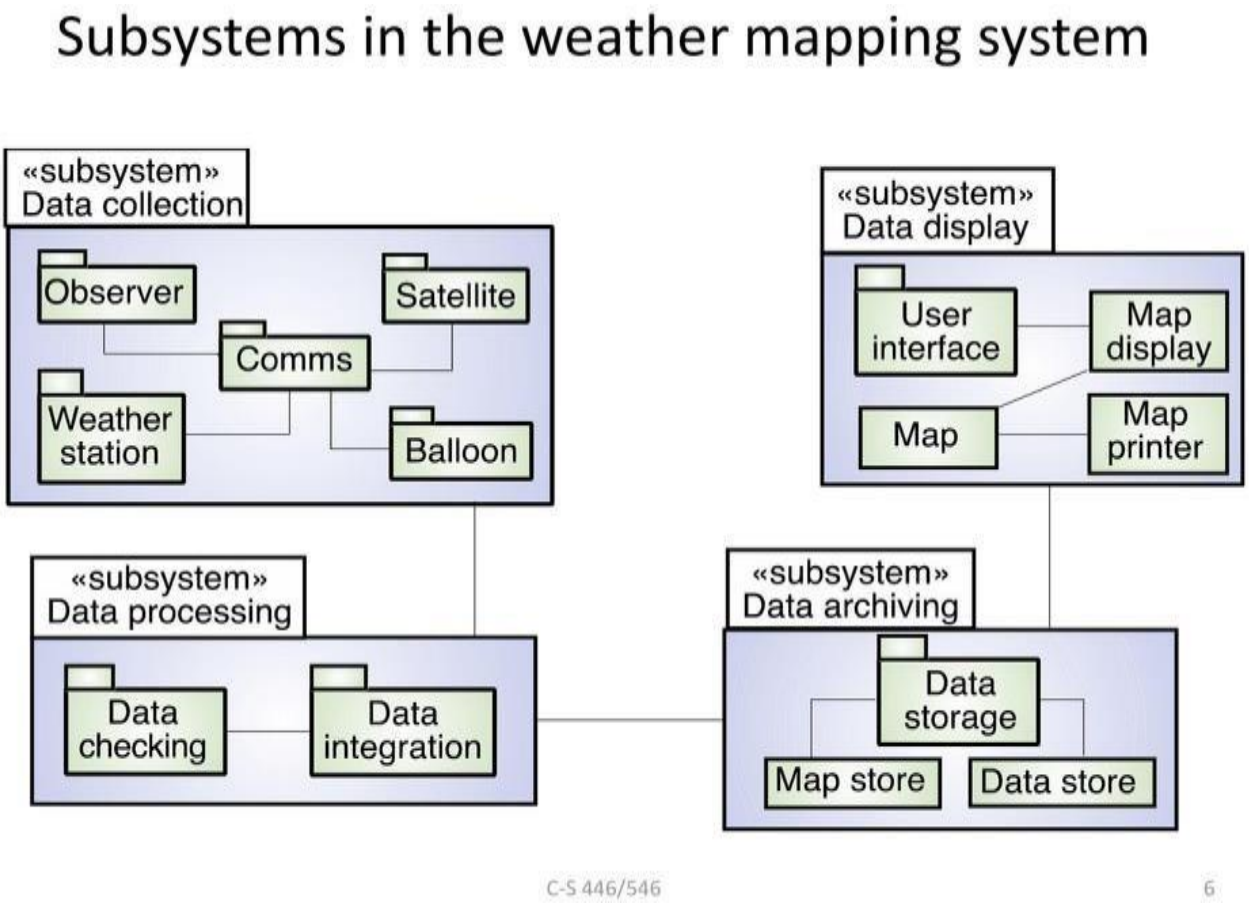
Specify object interfaces.

A weather mapping system weather mapping system is required to generate weather maps on is required to generate weather maps on a regular basis using data collected

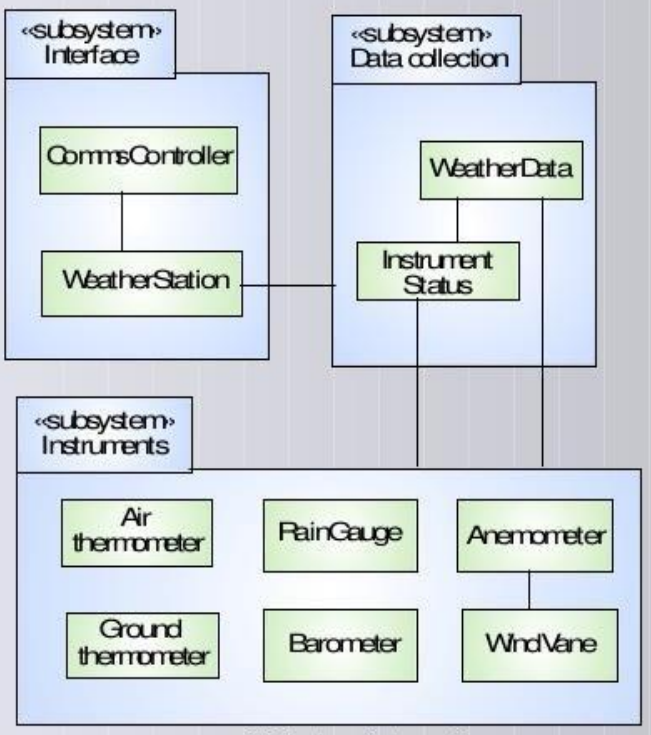
from remote, unattended weather stations and other data sources such as weather observers, balloons and satellites.

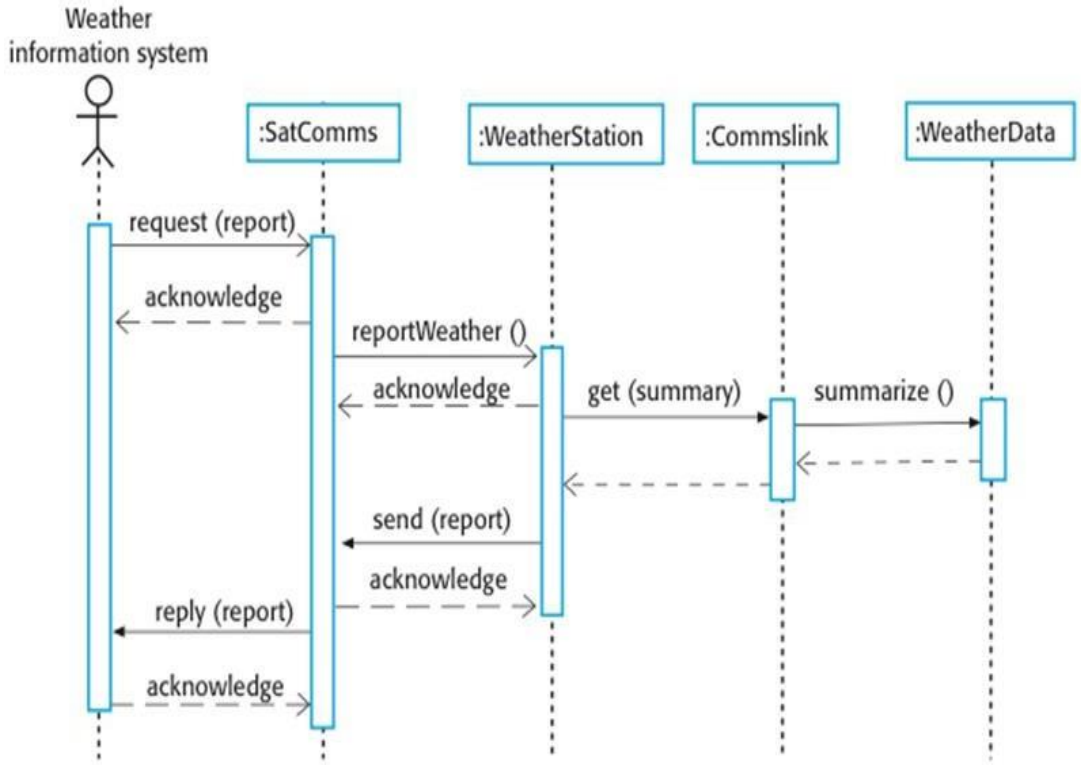
Weather stations transmit their data to the balloons and satellites. Weather stations transmit their data to the area computer in response to a request from that machine. The area computer system validates the collected data and integrates it with the data from different sources.

The integrated data is archived and, using data from this archive and a digitized map database a set of local weather maps is created.



Weather station subsystems




Weather
information system
:SatComms
:WeatherStation
:Commslink
:WeatherData
request (report)
acknowledge
reportWeather ()
acknowledge
get (summary)
summarize ()
send (report)
acknowledge
reply (report)
acknowledge

Chapter- 7 and 8 User Interface Design

1) What are the characteristics of GUI?

Windows
Icons
Menus
Pointing
Graphics.

2) What is user interface design?

User interface is the front-end application view to which user interacts in order to use the software. User can manipulate and control the software as well as hardware by means of user interface.

3) Name the categories of UI?

Command Line Interface

Graphical User Interface

4) Name different types of user system interaction?

Command language
Formfillin
Menu selection
Direct Manipulation

5) What is command line Interface?

CLI is minimum interface a software can provide to its users.

CLI provides a command prompt, the place where the user types the command and feeds to the system.

6) What are the components of GUI?

Text-Box - Provides an area for user to type and enter text-based data.

- **Buttons** - They imitate real life buttons and are used to submit inputs to the software.
- **Radio-button** - Displays available options for selection. Only one can be selected among all offered.
- **Check-box** - Functions similar to list-box. When an option is selected, the box is marked as checked. Multiple options represented by check boxes can be selected.
- **List-box** - Provides list of available items for selection. More than one item can be selected.

7) Explain GUI design Activity?

There are a number of activities performed for designing user interface. The process of GUI design and implementation is alike SDLC. Any model can be used for GUI implementation among Waterfall, Iterative or Spiral Model.

- **GUI Requirement Gathering** - The designers may like to have list of all functional and non-functional requirements of GUI. This can be taken from user and their existing software solution.
- **User Analysis** - The designer studies who is going to use the software GUI. The target audience matters as the design details change according to the knowledge and competency level of the user. If user is technical savvy, advanced and complex GUI can be incorporated. For a novice user, more information is included on how-to of software.
- **Task Analysis** - Designers have to analyze what task is to be done by the software solution. Here in GUI, it does not matter how it will be done. Tasks can be representedin hierarchical manner taking one major task and dividing it further into smaller sub- tasks. Tasks provide goals for GUI presentation. Flow of information among sub-tasks determines the flow of GUI contents in the software.
- **GUI Design & implementation** - Designers after having information about requirements, tasks and user environment, design the GUI and implements into code andembed the GUI with working or dummy software in the background. It is then self- tested by the developers.
- **Testing** - GUI testing can be done in various ways. Organization can have in-house inspection, direct involvement of users and release of beta version are few of them. Testing may include usability, compatibility, user acceptance etc.

8). What are the characteristics of GUI?

GUI Characteristics

Characteristics	Descriptions
Windows	**Multiple windows allow different information to be displayed simultaneously on the user's screen.**
Icons	**Icons different types of information. On some systems, icons represent files. On other icons describes processes.**
Menus	**Commands are selected from a menu rather than typed in a command language.**

Pointing	**A pointing device such as a mouse is used for selecting choices from a menu or indicating items of interests in a window.**
Graphics	**Graphics elements can be mixed with text or the same display.**

Chapter 8 Reliability and Reusability

1). Define Reliability?

Software Reliability means Operational reliability. It is described as the ability of a system or component to perform its required functions under static conditions for a specific period.

2). Define S/W Relaibility and H/W Reliability?

Software Reliability is the probability of failure-free software operation for a specified period of time in a specified environment. Software Reliability is also an important factor affecting system reliability.

Reliability of software is maintained until any fault find in hardware which affects the path of the data. Reliability of a software refers to the probability of a failing hardware part and the calculation of time for the maintenance of hardware parts. Physically errors always find in software system.

3). What is Software Reliability metrics?

Some reliability metrics which can be used to quantify the reliability of the software product are as follows: Mean Time to Failure (MTTF) Mean Time to Repair (MTTR) Mean Time Between Failure (MTBR) Rate of occurrence of failure (ROCOF) Probability of Failure on Demand (POFOD) Availability (AVAIL)

4). Define MTTF?

Mean time to repair (MTTR): The average time to repair and restore a failed system. It's a measure of the maintainability of a repairable component or service. Depending on the complexity of the device and the associated issue, MTTR can be measured in minutes, hours or days.

5). Define MTTF?

Mean time to failure (MTTF) is a maintenance metric that measures the average amount of time a non-repairable asset operates before it fails. Because MTTF is relevant only for assets and equipment that cannot or should not be repaired, MTTF can also be thought of as the average lifespan of an asset.

6). What is MTBF?

Software availability is measured in terms of mean time between failures(MTBF). MTBF consists of mean time to failure (MTTF) and mean time to repair (MTTR). MTTF is the difference of time between two consecutive failures and MTTR is the time required to fix the failure.

7). What is POFOD?

POFOD is described as the probability that the system will fail when a service is requested. It is the number of system deficiency given several systems inputs. POFOD is the possibility that the system will fail when a service request is made.

8) . Define AVAIL?

Availability is defined as the probability that the system is operating properly when it is requested for use. In other words, availability is the probability that a system is not failed or undergoing a repair action when it needs to be used.

9) . Write short notes on statistical testing?

Statistical Testing is a testing method whose objective is to work out the undependable of software package product instead of discovering errors. check cases ar designed for applied mathematics testing with a wholly totally different objective than those of typical testing.

Steps in Statistical Testing:

Statistical testing permits one to focus on testing those elements of the system that ar presumably to be used. the primary step of applied mathematics testing is to work outthe operation profile of the software package. a successive step is to get a group of check knowledge reminiscent of the determined operation profile. The third step is touse the check cases to the software package and record the time between every failure. once a statistically important range of failures are ascertained, the undependable may be computed.

10). What are the advantages of S/W reuse?

Increase **software** productivity.

Shorten **software** development time.

Improve **software** system interoperability.

Develop **software with** fewer people.

Move personnel more easily from project to project.

Reduce **software** development and maintenance costs.

Produce more standardized **software**.

Define Failure?

A **failure** is the inability of a **software** system or component to perform its required functions within specified performance requirements. When a defect reaches the end customer it is called a **Failure**. During development **Failures** are usually observed by testers

2)Define Fault and Bug?

Software fault is also known as **defect**, arises when the expected result don't match with the actual results. It can also be **error**, flaw, failure, or **fault** in a computer program.

3) Define Bug?

A bug is the result of a coding error. An Error found in the development environment before the product is shipped to the customer. A programming error that causes a program to work poorly, produce incorrect results or crash. An error in software or hardware that causes a program to malfunction. Bug is terminology of Tester.

4) Define Verification?

Verification is the process of evaluating work-products of a development phase to determine whether they meet the specified requirements. **verification** ensures that the product is built according to the requirements and design specifications.

5) Define validation?

Validation is the process of checking whether the **software** product is up to the mark or in other words product has high level requirements. It is the process of checking the **validation** ofproduct i.e. it checks what we are developing is the right product.

6). Define Test case?

A **test case** is a document, which has a set of **test** data, preconditions, expected results and postconditions, developed for a particular **test** scenario in order to verify compliance against a specific requirement.

7). Define Test Plan?

Test planning, the most important activity to ensure that there is initially a list of tasks and milestones in a baseline plan to track the progress of the project. It also defines the size of the test effort.

It is the main document often called as master test plan or a project test plan and usually developed during the early phase of the project.

8). **Explain test plan activity?**

- To determine the scope and the risks that need to be tested and that are NOT to be tested.
- Documenting Test Strategy.
- Making sure that the testing activities have been included.
- Deciding Entry and Exit criteria.
- Evaluating the test estimate.

- Planning when and how to test and deciding how the test results will be evaluated, and defining test exit criterion.
- The Test artefacts delivered as part of test execution.
- Defining the management information, including the metrics required and defect resolution and risk issues.
- Ensuring that the test documentation generates repeatable test assets.

9). **Explain Black Box Testing?**

Black-box testing is a method of software testing that examines the functionality of an application based on the specifications. It is also known as Specifications based testing. Independent Testing Team usually performs this type of testing during the software testing life cycle.

Black Box Testing Techniques

Syntax Driven Testing – This type of testing is applied to systems that can be syntactically represented by some language. For example- compilers,language that can be represented by context free grammar. In this, the test cases are generated so that each grammar rule is used at least once.

2. Equivalence partitioning – It is often seen that many type of inputs work similarly so instead of giving all of them separately we can group them together and test only one input of each group. The idea is to partition the input domain of the system into a number of equivalence classes such that each member of class works in a similar way, i.e., if a test case in one class results in some error, other members of class would also result into same error.

The technique involves two steps:

1. **Identification of equivalence class –** Partition any input domain into minimum two sets: **valid values** and **invalid values**. For example, if the valid range is 0 to 100 then select one valid input like 49 and one invalid like 104.
2. **Generating test cases –**
   (i) To each valid and invalid class of input assign unique identification number.
   (ii) Write test case covering all valid and invalid test case considering that no two invalid inputs mask each other.

To calculate the square root of a number, the equivalence classes will be:

(a) Valid inputs:

- Whole number which is a perfect square- output will be an integer.
- Whole number which is not a perfect square- output will be decimal number.
- Positive decimals

(b) Invalid inputs:

- Negative numbers(integer or decimal).
- Characters other that numbers like “a”,”!”,”;”,etc.

3. **Boundary value analysis –** Boundaries are very good places for errors to occur. Hence if test cases are designed for boundary values of input domain then the efficiency of testing

improves and probability of finding errors also increase. For example – If valid range is 10 to 100 then test for 10,100 also apart from valid and invalid inputs.
4. **Cause effect Graphing –** This technique establishes relationship between logical input called causes with corresponding actions called effect. The causes and effects are represented using Boolean graphs. The following steps are followed:
1. Identify inputs (causes) and outputs (effect).
2. Develop cause effect graph.
3. Transform the graph into decision table.
4. Convert decision table rules to test cases.

10) Explain white box testing?

White Box Testing is software testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability andsecurity. In white box testing, code is visible to testers so it is also called Clear box testing, Openbox testing, Transparent box testing, Code-based testing and Glass box testing.

11) . Explain white box testing techniques?

A major White box testing technique is Code Coverage analysis. Code Coverage analysis eliminates gaps in a Test Case suite. It identifies areas of a program that are not exercised by a set of test cases.

Statement Coverage:- This technique requires every possible statement in the code to be tested at least once during the testing process of software engineering.

Branch Coverage - This technique checks every possible path (if-else and other conditional loops) of a software application.

- Condition Coverage
- Multiple Condition Coverage
- Finite State Machine Coverage
- Path Coverage
- Control flow testing
- Data flow testing

12 Explain Interface testing?

Interface Testing is defined as a software testing type which verifies whether the communication between two different software systems is done correctly.

A connection that integrates two components is called interface.

13). Explain clean room software development process?

Formal specification:

The computer code to be developed is formally given. A state-transition model that shows system responses to stimuli is employed to precise the specification.

Incremental development:
The computer code is partitioned off into increments that area unit developed and valid on individual basis mistreatment the white room method.

Structured programming:
Only a restricted range of management and information abstraction constructs area unit used. The program development method is that the method of stepwise refinement of the specification.

Static verification:
The developed computer code is statically verified mistreatment rigorous computer code inspections. there's no unit or module testing method for code parts.

Statistical testing of the system:
The integrated computer code increment is tested statistically to work out its responsibility. These applied mathematics tests area unit supported the operational profile that is developed in parallel with the system specification.

14). Explain Program Analysis Tool?

Program Analysis Tool is an automated tool whose input is the source code or the executable code of a program and the output is the observation of characteristics of the program. It gives various characteristics of the program such as its size, complexity, adequacy of commenting, adherence to programming standards and many other characteristics.

Static Program Analysis Tool is such a program analysis tool that evaluates and computes various characteristics of a software product without executing it. Normally, static program analysis tools analyze some structural representation of a program to reach a certain analytical conclusion. Basically some structural properties are analyzed using static program analysis tools.

1. Whether the coding standards have been fulfilled or not.
2. Some programming errors such as uninitialized variables.
3. Mismatch between actual and formal parameters.
4. Variables that are declared but never used.

Dynamic Program Analysis Tools:
Dynamic Program Analysis Tool is such type of program analysis tool that require the program to be executed and its actual behavior to be observed. A dynamic program analyzer basically implements the code. It adds additional statements in the source code to collect the traces of program execution. When the code is executed, it allows us to observe the behavior of the software for different test cases.

Once the software is tested and its behavior is observed, the dynamic program analysis tool performs a post execution analysis and produces reports which describe the structural coverage that has been achieved by the complete testing process for the program.
For example, the post execution dynamic analysis report may provide data on extent statement, branch and path coverage achieved.

The results of dynamic program analysis tools are in the form of a histogram or a pie chart. It describes the structural coverage obtained for different modules of the program. The output ofa dynamic program analysis tool can be stored and printed easily and provides evidence that complete testing has been done. The result of dynamic analysis is the extent of testing performed as white box testing. If the testing result is not satisfactory then more test cases are designed and added to the test scenario. Also dynamic analysis helps in elimination of redundant test cases.