UNIT 1

INTRODUCTION TO SOFTWARE ENGINEERING, A GENERIC VIEW OF PROCESS, PROCESS MODELS

PART-A
SHORT QUESTIONS WITH SOLUTIONS

Q1. Define the term software and software engineering.

Answer :

Model Paper-I, Q1(a)

Software

In general terms, software is referred as an organized set of instructions which when executed by means of a given computing device delivers the desired result by considering various processes and functions.

It is an organized set of instructions capable of accepting inputs, processes them and delivers the result in the form of functions and performs as expected by the user. Apart from this, it refers to the records (say software manuals) which helps and guides the users to efficiently deal with it. It is now-a-days delivered in the form of a package accumulating the design documents, source code, installations implementation manuals, operations system manuals.

Software Engineering

Software engineering is defined as establishment and application of sound engineering principles for obtaining an economically feasible and reliable software that can run efficiently on any real-time machine.

Q2. Write short notes on any two software applications.

Answer :

The two software applications are as follows,

(a) Application Software

Application software usually resides on a single system which is capable of satisfying only business requirements and involves management/technical decision making.

(b) Netsourcing

With an unpredictable growth of the internet, the software engineers are now forced to look forward for the development of simple as well as more sophisticated softwares so that, it is the end user who can take larger benefits from such applications, as internet in today's world, is not only acting like an engine but also as the major source for obtaining large volumes of data.

Q3. Explain software crisis.

Answer :

Nov./Dec.-17(R15), Q1(b)

Software crisis refers to critical point which is faced by software developers at the time of software development. The crucial point may be due to low quality, high cost, incompatibility, presence of errors, low productivity, less efficiency in tools and methods adopted.

Q4. What is legacy software? Explain.

Answer : (Model Paper-III, Q1(b) | Nov./Dec.-16(R13), Q1(a))

Legacy Software can be defined as an old software developed in the past. This software is still being used in the present era as it performs essential business activities. It may include procedure which are no longer relevant in the newer computing environment. As business requirements are dynamic, the legacy software system undergo continuous modifications so as to make the software adaptable to the new business requirements and to make it interoperable with the current computing environments.

Though, legacy software systems are becoming problematic in large organizations because of their high maintenance cost, they are still being used in the organizations due to the difficulties and risks encountered while replacing these systems with modern systems. Legacy systems are supportive to essential business activities and therefore they are considered as business critical system.

Q5. Give any three types of changes made to legacy system.

Answer :

The changes made to legacy system are as follows,

(i) Making the Software Adaptable

The software can meet the requirements of new computing environments, by making the software adaptable to the computing environment.

(ii) Enhancing the Software

The characteristic features of software needs to be enhanced so that the software can easily implement the new business requirements.

(iii) Extending the Software

The software needs to be extended so that it can achieve the interoperability feature.

Q6. List and define the various software myths.

Answer : Model Paper-II, Q1(a)

The three software myths are,

(i) Management Myths

These are the myths believed by software managers who are responsible for improving quality and controlling budgets.

(ii) Customer Myths

These are the myths believed by customers who can either be internal (technical group, marketing/sales department) or external to an organization.

(iii) Practitioner's Myths

Practitioner's myths are misconceptions believed by many software practitioners.

Q7. Discuss the objective of CASE.

Answer :

CASE stands for Computer Aided Software Engineering. It can be defined as a software or a technology, whose objectives are,

(a) To provide support to software process by implementing automation on the activities of software process like design, programming etc.

(b) To give information regarding the development of software.

Q8. What is CMM?

Answer :

Capability Maturity Model (CMM) is a maturity framework strategy, that focuses on continuously improving the management and development of the organizational workforce. CMM provides the organization with the basic requirements, for building of the software process. It provides an evolutionary path from an inconsistent organizational practices, to a highly disciplined developmental practice, which helps to improve the knowledge, skills and development of the software process.

Q9. Discuss the major problems with the capability maturity model.

Answer :

CMM provides various advantages to the develo-pment of software process, even though some of the disadvantages of CMM model arise in the development of software process. They are as follows,

1. The CMM model does not allow risk analysis and management as the key process areas. Whereas, the risk management is a process that helps to identify the potential risks in the software development process.
2. The CMM model does not concentrate on product development, rather it concentrates on project management which does not allow an organization to utilize the technologies such as structured methods, prototyping and tools for static analysis.
3. The CMM model has complicated rules and procedures for small organizations.

The functionalities of this model are not defined. Only few organizations can use this model and on other hand they do not specify for which kind of organization this model will be suitable. Using this type of model leads to the drawback of the software development. It will be complicated to use this model in small organizations.

Q10. List the sources of software standards.

Answer :

The sources of software standards include different organisations. They are,

1. ISO (International Standards Organization)
2. IEEE (Institute of Electrical and Electronic Engineers)
3. ANSI (American National Standards Institute)
4. DOD (U.S Department of Defense).
5. IEE (Institute of Electrical Engineers in UK) and BS (British Standard Institute)
6. OMG (Object Management Group).

Q11. What is PSP?

Answer :

Every team member possesses his own personal software process. If this personal process remains ineffective, then he/she is advised to travel through the four phases of software development, each time being trained thoroughly and maturing himself, so that he can remain in a position to analyze the project development, at the same time the quality of it. But, this session does not end here, rather PSP make an individual to get involved in the project planning and also in maintaining the quality of all the software products on a whole. In order to achieve this, PSP adheres following five major framework activities,

- **Planning**

  In this activity, the user we initially look forward for the requirements of the project. Later the vision is extended on the defect estimates. Finally, by estimating the development task and documenting them, this session is concluded.

- **High Level Design**

  In this case, we usually address the scenario of entire project by developing certain high level designs. If uncertainty prevails in such diagrams, we look forward for prototypes. By documenting the whole scenario the current session is concluded.

Q12. What is waterfall model?

Answer : **Model Paper-III, Q1(a)**

Waterfall model remains one of the oldest strategies ever applied, in the development of a software. It is also known as *classic life cycle model*, which divides the entire software development process into five main phases. Following is the diagrammatical representation of waterfall model,

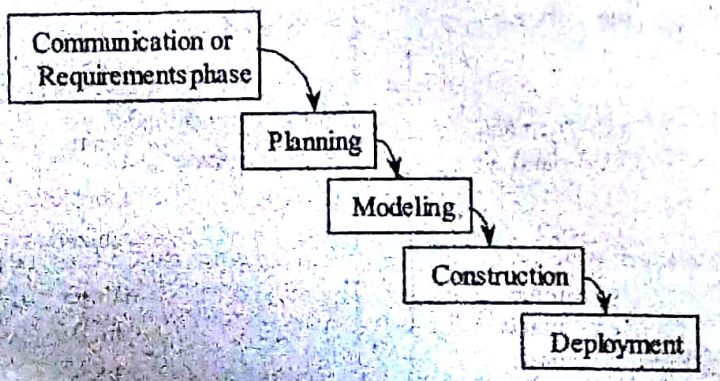


Figure: Phases of Waterfall Model

The waterfall model was proposed with feedback loops. However, most of the organizations avoid these loops. Thus this model is also called Linear Sequential Model.

Q13. List the merits and demerits of RAD process model.

Answer :

Merits

1. Software projects are given a deadline of less than three months can opt for RAD. Since, each RAD team handles a mo function which are then integrated to form a complete system.
2. The task is divided among teams to fasten the development process.

Demerits

1. Large but scalable projects using RAD development model need a huge number of human resource in order to form pro RAD teams.
2. Both customers and developers must commit to the rapid-fire activities to be performed to finish the task. Else, the def deadline is not met and the project fails.
3. Improper system moduralization makes it difficult for building RAD components.
4. RAD is not applicable to systems wherein performance is achieved by tuning interfaces to system components.
5. High technical risks prohibits the usage of RAD.

Q14. What are the advantages of prototyping model over waterfall model?

Model Paper-I, Q

Answer :

The advantages of prototyping model over waterfall model are as follows,

(i) The Prototype Models give a prototype that can be used to illustrate input data formats, messages, reports and interacti dialogues for the customers. This is a valuable mechanism for explaining various processing options to the customer an for gaining a better understanding of the customer's needs. This is not possible in Waterfall Model.

(ii) The prototype explores technical issues in the proposed product. Often, a major design decision will depend on, say, th response time of a device controller or the efficiency of a sorting algorithm. In these cases, a prototype may be the bes or only, way to resolve the issue. The waterfall model does not explore the technical issues in the proposed product.

Q15. Differentiate between waterfall and incremental process models.

Answer :

May/June-12, Set-4, Q3

	Waterfall Model		Incremental Model
1.	The waterfall model is called a linear sequential life cycle model as it develops the software sequentially.	1.	The incremental model is iterative. It develops the software in multiple iterations.
2.	In waterfall model, the software is developed in sequence and delivered at the end at once.	2.	In incremental model, the software is developed in increments and each phase is delivered separately at successive points of time.
3.	Requirements cannot be changed once fixed.	3.	Requirements can be changed after every increment.
4.	Risks are high in waterfall model and cannot be analyzed before the last phase.	4.	Risks are identified and solved after every iteration.
5.	Changing the requirements becomes costly as all the phases from the beginning have to be repeated.	5.	Changing the requirements in the incremental model is easy, as new features can be added after every iteration.

Q16. Compare the linear sequential model with the spiral model.

(Model Paper-II, Q1(b) | Nov./Dec.-12 (R09), Q2(a))

Answer :

	Linear Sequential (Waterfall Model)		Spiral Model
1.	The flow from one phase to another phase is in linear fashion.	1.	The flow from one phase to another phase is in iterations (spirals).
2.	Changes are very difficult to implement.	2.	Changes can be implemented easily.
3.	Fixing of errors is difficult.	3.	Fixing of errors is relatively easy.
4.	Easy to understand and adapt.	4.	Relatively difficult to understand and adapt.
5.	Risk management is difficult.	5.	Better risk management.
6.	Suitable for applications that need development from scratch.	6.	Suitable for component based development.
7.	It is used for small scale systems.	7.	It is used for large scale systems.
8.	It is economical to adopt.	8.	It is expensive to adopt.

Q17. What are the advantages of unified process?

Nov./Dec.-16(R13), Q1(b)

Answer :

The advantages of unified process are as follows,

1. It covers the complete software development life cycle.
2. Even though the unified process model came into existence after a true hardwork of over 20 years, it is available on internet in the form of an electronic guide (available to everyone located anywhere around the globe).
3. Most of the modern techniques and approaches use the unified process model as a set of guidelines.
4. The best practices for software development are supported by unified process model.
5. Unified process model does not result in a frozen product. Rather the product is ever evolving and is constantly maintained.
6. Its process architecture can be tailored as per requirement.
7. It encourages UML as the best process-oriented languages protocols.

Q18. What is an agile process? Explain.

Nov.-15(R13), Q1(a)

Answer :

Agile processes are the processes that are evolved in response to the documentation and bureaucracy-based processes like the waterfall model. Depending upon the following principles, agile approaches build an optimum project.

1. An efficient software model must be chosen for the progress of a project.
2. The software must be build and delivered quickly in small iterations.
3. The software must be capable of accepting the changes in requirements even in the later stages of software development.
4. Face-to-face communication between the clients or customers must be done rather than preparing the documentation.
5. The customers involvement and their continuous feedbacks about the software is essential for producing high quality software.
6. A simple design that evolves and improves regularly over a period of time must be considered for managing the different scenarios rather than the detailed design.
7. The empowered development teams must decide the delivery time of the software.

UNIT 2

SOFTWARE REQUIREMENTS, REQUIREMENTS ENGINEERING PROCESS, SYSTEM MODELS

SIA GROUP

PART-A
SHORT QUESTIONS WITH SOLUTIONS

Q1. Write any three metrics defined to measure non-functional requirements.

Answer :

Model Paper-I, Q1(c)

The metrics defined to measure non-functional requirements are as follows,

- Property : Speed
  Metric : It is measured in terms of screen refresh time, user or event response time, number of transactions being processed per second.
- Property : Portability
  Metric : (i) Percentage of statements directly dependent on the target system.
  (ii) Total number of target systems.
- Property : Ease of use
  Metric : (i) Total number of help frames required.
  (ii) Time spent in training.

Q2. What are the steps to be followed while performing the domain analysis process?

Answer :

Model Paper-II, Q1(c)

The following are the steps carried out while performing the domain analysis process,

(i) The domain that needs to be analyzed must be defined.
(ii) The items retrieved from sources of domain knowledge must be categorized.
(iii) The sample of application defined in the domain must be collected.
(iv) Every individual application in the sample is analyzed and based on this, analysis class must be defined.
(v) An analysis model for the class must be developed.

Q3. What are the characteristics of a good SRS document?

Answer :

Nov.-15(R13), Q1(d)

There are several requirements and properties to be satisfied by the SRS to satisfy the basic goals of the proposed system. The desirable characteristics are,

1. Correct
2. Complete
3. Unambiguous
4. Verifiable
5. Consistent
6. Stability
7. Modifiable/Convertible
8. Traceable.

Q4. What is the intent of requirements validation?

Answer : (Model Paper-III, Q1(d) | Nov.-15(R13), Q1(c))

This is the final step in the requirements engineering process, where all the proposed and refined requirements are verified in order to conclude that, they are laid in accordance to the user specifications. Hence, the end product of this phase will be the requirements documents.

Hence, in short all the above mentioned activities can be partitioned under three activities.

- Discovery
- Documentation and
- Checking.

However, when the developer visualize the practical conditions the scenario changes drastically, human requirements change depending on the time factor, the organization purchasing the software changes, the hardware on which the given software being implemented changes etc. Hence, the software developed should be well acquainted to satisfy all these anomalies.

Q5. What is the significance of feasibility study?

Answer : Nov./Dec.-16(R13), Q1(d)

The feasibility study forms the initial step in the requirements engineering process. The developers can start with feasibility study process whenever the information related to the following aspects are obtained.

- The business requirements.
- The extent to which current system is capable of satisfying the requirement.
- A brief description of the system.

Hence, whenever feasibility study process, is completed, the developers will be ready with the documents specifying whether the current system is capable/incapable of satisfying complete business requirements. Based on the result, the decision on whether to set aside, move further or refine the current system is taken. Therefore, when the feasibility study reports are developed, it should contain data related to the following aspects.

- Does the current report satisfy the business aspects of the organization involved in developing the system?
- Is it possible to implement the system using the current technology within given cost and schedule constraints?
- Can system be integrated with certain other well defined systems?

Q6. Define velwpoints.

Answer :

In technical aspects, the sources which remain information providers in the requirements discovery phase are often referred to as "viewpoints". Each of these viewpoints provide a part of information for requirements discovery. The information gained from a given viewpoint may match with the information acquired from other viewpoints.

Q7. Write short notes on VORD method.

Answer :

The Viewpoint Oriented Requirement Definition (VORD) is a method of requirements elicitation and analysis proposed by Kotonya and Sommerville. It has been designed as a service framework for requirements discovery and analysis. It also includes different steps in order to translate the analysis into an object-oriented system model. A viewpoint-oriented analysis has a major ability to identify several perspectives and provides a framework for discovering process in the requirements designed by various stakeholders.

Q8. What do you mean by viewpoint structuring? Give an example for it.

Answer : Model Paper-II, Q1(d)

Viewpoint structuring is an important phase of VORD method, which involves combining of similar view points into a hierarchical form. The similar viewpoints based on the specific services are arranged in a structural form for easy understanding and analysis purposes.

In technical aspect, viewpoint structuring is an iterative process of dividing the target system into a hierarchy of functional sub systems. It provides structure to the viewpoints, which are placed at the same level as the target system. Therefore, each functional subsystem can be represented as a viewpoint. Viewpoint structuring analyzes the target system as well as the functional sub systems in order to understand the requirements of the software system.

A simple example of viewpoint structuring of a college is shown below,

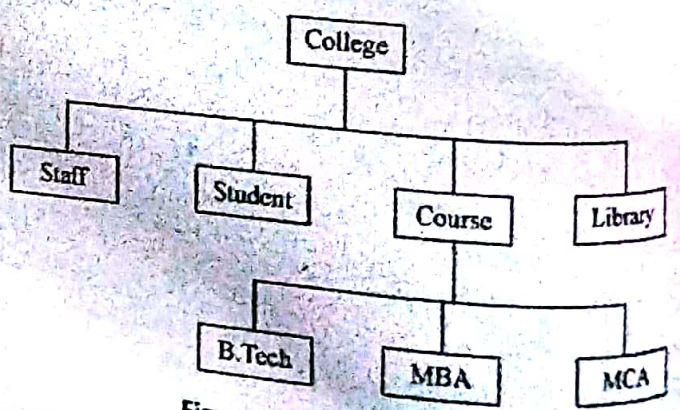


Figure: Viewpoint Structuring

Q9. Define Brainstorming.

Answer :

Brainstorming is defined as a group creativity technique that is designed to generate a large number of ideas as an optimal solution to a problem. During requirements elicitation and analysis, brainstorming may be used to understand the requirements in an effective and efficient manner. It is particularly helpful when a problem with many issues arises, so that generating new ideas, creative thinking and new opportunities will enable to develop highly creative solution to a problem.

Q10. Define use cases.

Answer : Model Paper-III, Q1(c)

Use case is the essential feature of the UML notation used in object oriented system models. Use cases acts as a scenario in the real world concept for describing the requirements. Analysis of a system use case is used by requirements engineers to describe the interaction of a user with the proposed software system that meet their requirements.

Q11. Discuss the statement, "requirements management needs automated support".

Answer :

Automated support is necessary for requirements management. The CASE tools that are required for this support must be selected during the planning phase. The automated tool must support the following functions,

(i) **Storage for Requirements**

The storage for requirements must be secured and well managed. Data should be easily accessible to all people involved in the requirements engineering process.

(ii) **Management for Traceability**

This feature allows the discovery of related requirements. Relationships between the requirements may be discovered using tools that use natural processing techniques.

(iii) **Management for Changes**

Change management process can be simplified with the help of available active tool support. Changes in the requirement can be documented and well managed by using automated tools.

Small projects does not require the use of special requirement management tools support can be provided hereby using available word processors, databases and spreads sheets. However, for large projects special requirements management tools are required.

Q12. What is ATC?

Answer :

Air Traffic Control (ATC) system is one which separates the air crafts in order to protect them from collisions, thereby organizing or stream lining flow of traffic. Here the process of preventing the collisions is called separation.

- When air craft flies with the assistance of air traffic control system, it is controlled in air space, otherwise uncontrolled air space. In controlled air space air traffic controllers are provided for separately guiding the air crafts. All pilots follow the instructions issued by ATC system.

Q13. Write a short note on open interviews.

Answer :

In this type of interview, the requirement engineering team discusses different issues with the stakeholders. The purpose of this interview is to gather significant information about the requirements of the stakeholders. In contrast to closed interviews, open interviews doesn't consists of any predefined set of questions.

Q14. What is volatile system requirements?

Answer :

Volatile system requirements are defined as unstable requirements which can be changed over a period of time. It is used when the system is developing or the system has started its operational process. These requirements are frequently changed based on the needs of a user or a software system.

For example, government releases the educational system policies that can be changed over a period of a time.

Q15. Define high level design matrix.

Answer : Model Paper-I, Q1(d)

A matrix that correlates any two documents requiring many-to-many relationship between them for determining the completeness of the relationship is called as high level design matrix. It is also known as traceability matrix or traceability table. It is used with high-level and detailed requirements of the software product that matches with the pacts of the high-level design, detailed design, test plan and test cases.

Q16. Write the purpose of context model.

Answer : Nov./Dec.-16(R13), Q1(c)

Context Model

In the process of software development we should decide on the system boundary and its environment at an early stage in the requirements elicitation and analysis process. Once we are done with this a simple architectural model is brought up.

Example

For example following is an architectural structure representing the context of Bank ATM system.

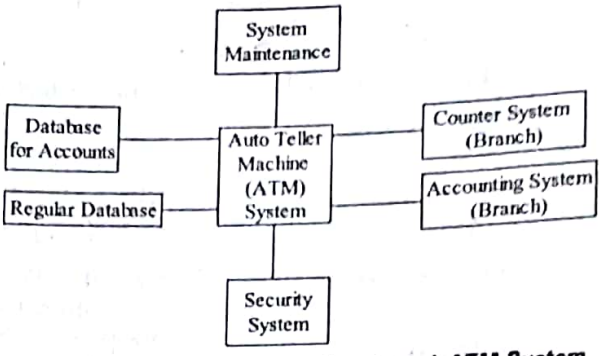


Figure: High Level Representation of Bank ATM System

The above figure represents only the block diagram which is nothing but a high level representation of the ATM system. In the figure, the rectangle represents specific subsystem and embedding the name of the subsystem within it. The lines connecting these rectangles represent the collaborations existing between these subsystem. However, the above figure depicts only the high level view of the system context. It does not express any details of various other independent systems working with it. Other external systems should be considered since they may produce or intake the data produced to or from the current system, there presence may have drastic impact on the current system, they may be directly connected to the current system etc.

Q17: Explain the need of behavioural models for software development.

Answer : Dec.-11, Set-1, Q5(a)

The behavioural model is needed in the software development process because,

- It dipicts the dynamic behaviour of the system.
- Its first type, data flow models, dipicts the data processing in the system where as its second type, state machine models, dipicts the reaction of the system towards the events.
- It represents the states of analysis classes and the system as a whole.
- It uses input from scenario, flow oriented and class based elements.
- It identifies the states, events and actions of a scenario.
- It indicates how a software responds to external events.
- It provides the facility of reviewing the accuracy and consistency of the system.

Q18. What is the use of state machine model?

Answer :

The state machine model is used to model the real time behaviour of a given system. It shows the states and events that result in change of the system's current states. However, it does not show how the data flow within the system. Hence, state machine model consists of states, events (external/internal) and transitions between these states.

Q19. Explain about data models.

Answer : Nov./Dec.-17(R15), Q1(d)

We know that, database forms one of the essential aspects of any of the systems being developed (irrespective of their sizes). Also we have models to represent the processing of logical data belonging to these systems and such models are often referred as semantic data models. Most of the software development organizations today rely on entity relation attribute modelling for modelling database systems. The main aspects of these models are the entities, attributes and their relationships that exists among these entities. This gained wider recognitions even in object oriented database modelling.

Q20. Explain the real life situations where waterfall model can be applied.

Answer : May/June-12, Set-4, Q3(a)

Automobile companies are one of the real life examples of waterfall model.

In the manufacturing of a car all its requirements are fixed already. With those requirements, the manufacturing process is initiated. After the completion of one phase, if the requirements change, then the process has to be backtracked to make necessary modifications in the requirements. This changes the entire system spending more time and money than before. These additional modifications add up to the disadvantages of waterfall model.

UNIT 3

DESIGN ENGINEERING, CREATING AN ARCHITECTURAL DESIGN, MODELING COMPONENT LEVEL DESIGN, PERFORMING USER INTERFACE DESIGN

PART-A
SHORT QUESTIONS WITH SOLUTIONS

Q1. How do we assess the quality of software design?

Answer :

Nov.-15(R13), Q1(f)

Design is a meaningful engineering representation of something that is to be built. It can be traced to a customer's requirements and at the same time assessed for quality against a set of predefined criteria for "good" design. In the software engineering context, design focuses on four major areas of concern, data, architecture, interfaces and components.

Software design is an iterative process through which requirements are translated into a "blueprint" for constructing the software.

The design specification (documentation) address different aspects of the design model and is completed as the designer refines his representation of the software. First, the overall scope of the design efforts is described.

Next, the data design is specified, database structure, any external file structures, internal data structures and a cross reference that connects data objects to specific files are all defined.

The architecture design indicates how the program architecture has been derived from the analysis model. In addition, structure charts are used to represent the module hierarchy (if applicable).

Q2. What do you mean by software design quality? Explain.

Answer :

(Model Paper-III, Q1(f) | Nov./Dec.-16(R13), Q1(f))

The quality of design that is under development can be assessed by conducting formal technical reviews or design walkthroughs . However, McGlaughin suggested three guidelines for the evaluation of a good design,

1. The requirements must be designed well. The design must include all the requirements of the customer. It must implement the explicit requirements of the analysis model.
2. The design must be properly understandable and readable by the testes who tests the software, as well as by the programmers who develops the code for the software.
3. The design must cover all the important points such as data, address, functions etc. A good software is designed for obtaining a good quality.

Q3. Give few characteristics of good design.

Answer :

Model Paper-II, Q1(e)

Following are few important guidelines often referred as characteristics of a good design.

1. The notion of design should reflect its meaning.
2. Design should be implemented based on the information obtained from the requirement analysis phase of the software development process.
3. The design should clearly represent the interfaces applicable in the system. This causes ease while providing connections between various modules of the software as well as between the software and its vicinity of implementation.
4. The design should consist of all the independent modules (i.e. the modules capable of functioning independently).
5. The design should exhibit clearly all the modules of the software.
6. Various elements such as data, architecture, interfaces and components, should be distinguished clearly.
7. Using the design, a software engineer should directly build data structure which is suitable for the implementation classes. Also the design should be the outcome of easily recognizable components.
8. A design should describe architecture, should encompass several components and should reflect dynamic behaviour etc.

Q4. Why should we not over modularize?

Answer :

Modularization is the process of dividing the software product into different modules. Before being assembled into an integrated product, each module is tested separately in order to improve the quality of software product. A module is simply a system component that provides one or more services to other modules for achieving the software improvements. But we should not over modularized because as many modules existed in the software system, the complexities can become more. Too many modules can lead to misunderstanding of software architecture.

Therefore, under modularity and over modularity should be avoided.

Q5. Define functional Independence.

Answer :

Model Paper-I, Q1(e)

Functional independence is an effective property of modularity, which relates to the concept of abstraction and information hiding. Functional independence is achieved by developing modules with the help of a unique function that does not need to interact with other modules. In simple terms, the software design for each module provides a specific sub-function of requirements that has a simple interface in order to view many parts of the program structure.

Independent modules are easier to maintain and test because the effects caused by design or code modification are limited, error propagation is reduced and reversible modules are possible. Functional independence can be measured using two qualitative criteria such as cohesion and coupling.

Q6. Explain the function independence consequences to software design.

Answer :

Dec.-11, Set-2, Q7(a)

The consequences of functional independence are given as follows,

(i) Error Isolation

Error propagation is reduced because the degree of interaction of an independent module with the other modules is less. Therefore, if any error is found in the existing module it will not directly affect the other modules.

(ii) Scope of Reuse

A module can be reused because the interaction of each module with other modules is minimal and each module has its own well-defined and precise functions. Such modules are called cohesive and they can be easily reused.

(iii) Understandability

The design complexity can be minimized because different modules have less interaction with other modules and can be understood independently.

Q7. Discuss the statement "abstraction and refinement are complementary concepts".

Answer :

Model Paper-III, Q1(e)

Abstraction and Refinement are Complimentary Concepts

The abstraction and refinement are closely related with one another.

Refinement

It is a process of elaboration. Modern software development is a complicated process especially when software ystem becomes large and complicated. Therefore, software developers must apply software refinement in order to proceed from higher levels of abstraction to a final executable software system by appending essential details.

Refinement provides even the low-level details as the (information) design proceeds with the progress in refinement every time it allows a designer to elaborate the original software development process and more and more details are added over time.

Abstraction

To increase the efficiency and to decrease the complexity in a software development process we makes use of abstraction.

Abstraction is defined as the process of hiding all the irrelevant data and provides only the essential or important data. It enables the designer to specify procedure and data thereby suppressing low-level details.

Q8. Write about any three types of design classes.

Answer :

The three different types of design classes are given below,

Process Classes

These classes represent the low-level graded business abstractions, for managing the business.

Persistent Classes

These classes are used to store large information (usually a database).

User Interface Classes

These classes are used to implement the abstractions for human computer interaction, usually they (HCI) provide the visual representation of the elements of the metaphor used in HCI.

Q9. Write short notes on architectural patterns.

Answer :

Architectural pattern is a bit analogous to architectural style. It transforms the design of an architecture. However, architecture pattern and architecture style differ with each other in number of ways.

- Pattern usually considers only single aspect of the entire architecture.
- It specifies certain rules which are to be obeyed by a given architecture. These rules define how the software will perform a part of its functionality at the infrastructure level.
- It exposes only certain behavioural facts of the architecture.

Architecture patterns can be combined with an architectural style in order to provide a suitable shape to the existing software structure.

Q10. Distinguish between classes and components.

Answer :

	Classes		Components
1.	Each class is associated with operations and attributes.	1.	Components are associated with only operations.
2.	A class is a collection of objects, sharing similar properties and behavior.	2.	A component is a collection of logical elements. These elements can be classes or any other collaborations.
3.	Classes can be used to represent logical abstractions.	3.	These are used to represent physical elements, generally made of bits (say. data file, documents etc.).

Q11. Draw a software architecture for a human resource system.

Answer :

Software Architecture of Human Resource System

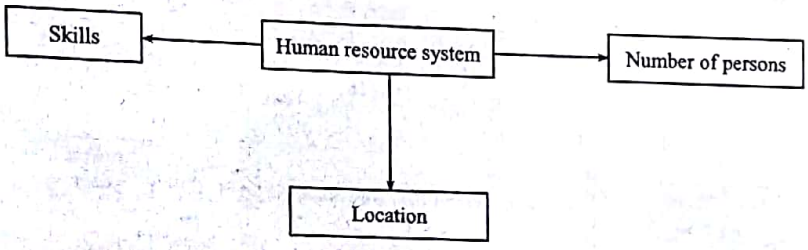


Figure: Software Architecture of Human Resource System

(a) Skills

Skills are the qualities that the planner selects for the system development.

(b) Location

Location plays a crucial role when the project is big and requires its project team members to be distributed across different locations to perform the software engineering tasks such as consulting with specialists etc.

(c) Number of Persons

It determines the number of persons required for the project.

Q12. Briefly discuss about the architectural description languages.

Answer :

Architectural Description Languages (ADL) is a formal approach, which is employed so as to provide specification about a software architecture. This specification contains syntax and semantics that provides detailed description regarding architectural design. The architectural description languages allow the designer to perform the following tasks,

(i) Decompose (break) the different architectural components.

(ii) Produce each component into a large architectural block.

(iii) Use interfaces as communication mechanism between the components.

The architectural design employing ADL helps in developing high effective evaluation methods for architecture during the evolution of the software design.

Examples of the architecural description languages includes Rapiche, Unicon, Assap, Wright, Aeme, UML.

Q13. Explain briefly data design elements.

Answer :

Data Design Elements

Data design often leads to a systematic representation of data in most abstract form. Later by applying refining tools the given data is progressed into a descent format which can be easily processed by any of the computer based system. While dealing with data design, one has to remember that the format (rather design) of data (on which we are currently working) exerts significant impact on the architecture of the software which processes it. This can be analyzed by considering the following stages of development.

(a) Component Level

At this level, we generally concentrate on the available forms of data like data structures and its corresponding algorithms. This turns out to be necessary, since quality of end product (software) depends primarily on these aspects at component level.

(b) Application Level

At this level, our attention shifts onto storage aspects i.e., converting the given data model into a database. This remains important to achieve the commercial objectives which were estimated on the product.

(c) Business or Commercial Level

Once the data is stored in the database, now we concentrate largely on improving its storage aspects i.e. acquiring the required data with less effort applied. This can be achieved by data warehousing mechanisms.

Q14. Draw the architectural diagram depicting the 'SafeHome' security system.

Answer :

The architectural diagram depicting the 'SafeHome' security system is given below,

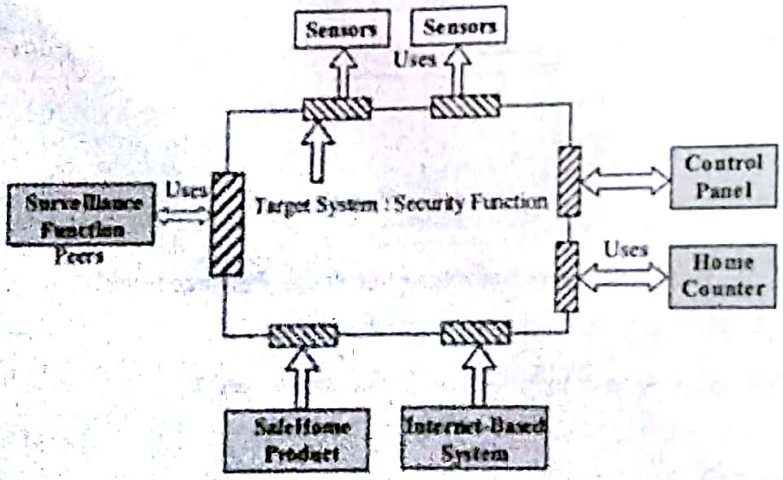


Figure: Architectural Diagram for SafeHome Security Systems

Q15. Differentiate between coupling and cohesion.

Answer :

(Model Paper-II, Q1(f) | Nov.-15(R13), Q1(e))

	Coupling		Cohesion
(i)	Coupling is the measure of interdependence among models.	(i)	Cohesion is the measure of functional grouping of elements within a single module.
(ii)	Modules are dependent on each other such that changes to one module affects other modules.	(ii)	Modules are dependent on each other such that changes to one module affects does not other modules directly.
(iii)	While designing a high degree of coupling must be avoided.	(iii)	While designing a high degree of cohesion must be considered.
(iv)	In high coupling, it is difficult to understand the modules.	(iv)	In high cohesion, it is easy to understand the modules.
(v)	When coupling is high, it is difficult to reuse the modules	(v)	When cohesion is high, it is easy to reuse the modules.

Q16. Differentiate between control and stamp coupling.

Answer :

	Control Coupling		Stamp Coupling
1.	Control coupling is said to occur when one module passes a variable to another module which control the internal logic of the other.	1.	Stamp coupling is said to occur when two modules pass data through a parameter which is a structure (or record).
2.	It is a moderate form of coupling.	2.	It is a loose form of coupling and simplifies interfaces between modules.
3.	The weakness of control coupling requires the calling program which receives the flags to have the knowledge of the internal logic.	3.	The weakness of stamp coupling is it encourages the creation of artificial data structures (i.e., binding un-related data elements into a structure).
4.	In control coupling, as rule of thumb one should use descriptive flags instead of control flags to describe the situation.	4.	In stamp coupling, as rule of thumb, one should pass a data structure containing required number of fields instead of passing a data structure with large number of fields.
5.	Example of control coupling is, print report (what-to-print-flag)	5.	Example of stamp coupling is, calc_order_amt (i.e., the amount of the order to be returned).

Q17. What is the use of interface analysis? Explain.

Answer :

Nov./Dec.-16(R13), Q1(e)

User interface analysis refers to the process of examining the requirements of interface to develop a good user interface. It is a central principle of all the software engineering process models that before attempting to find the solution to a problem, it is better to analyze and understand it problem includes understanding,

1. The end users
2. The tasks of end users
3. The content of the interface and
4. The environment in which the task will be performed.

Q18. What are the golden rules for user interface design?

Answer :

Nov./Dec.-17(R15), Q1(e)

The golden rules for user interface design are given as follows,

1. "Place the User in Control"
2. "Allow user interaction to be interruptible and undoable"
3. "Hide technical internals from the casual user"
4. "Design for direct interaction with objects that appear on the screen"
5. "Provide for flexible interaction"
6. "Define interaction modes in a way that does not force the user into unnecessary or undesired action"
7. "Streamline interaction as skills levels advance and allow the interaction to be customized".

Q19. What is cyclomatic complexity? What is its purpose?

Answer :

Nov.-15(R13), Q1(g)

Cyclomatic Complexity

Cyclomatic complexity is also known as conditional complexity. It is a metric used for computing the complexity of source code and for counting the number of linearly independent paths that comprise the program (or flow graph). The general formula to compute cyclomatic complexity is,

$$M = E - N + 2P$$

Where 'M' is the cyclomatic complexity, 'E' is the number of edges in the flowgraph, flowgraph and 'P' is the number of connected components.

This formula is applicable to flowgraphs that do not contain any disconnected components.

The cyclomatic complexity value assists the tester in number of ways.

(i) It is used to develop white box test cases.

(ii) It is used for computing the amount of time and resources, which are required during testing of software module.

(iii) It provides an estimation of testability of a module.

(iv) It gives an estimation of possible number of test cases.

Q20. What is the intent of information hiding?

Answer :

The intent of information hiding is to provide the mechanism for hiding the details or descriptions of the data structures, procedural processing and implementations. The users should not be aware of the information or knowledge of the module details because it is unnecessary information for the users that leads to errors. Information hiding is needed for modifications during the software testing and maintenance because most of the complex data and procedures need to be hidden from different parts of the software system.

Q21. Explain briefly about the framework.

Model Paper-I, Q11)

Answer :

Framework can be defined as the implementation of specific infrastructure using which a designer selects "reusable mini architecture". This architecture helps the designer by providing information about the generic structure and behavior associated with class software abstractions.

This information is provided together with a content that specifies the collaboration as well as the purpose of generic structure and behavior in a specific domain.

A framework is not considered as architectural pattern, instead it is considered as a structure consisting of set of "plugpoints". These plugpoints enables the framework to be adaptive to any problem domain. In addition to this, the plug points, allows a designer to combine problem specific classes or functionality within the structure.

Framework is said to be effective, only when they are applied without making any sort of changes. However, extending the framework by adding more design elements doesn't have any impact on the effectiveness of the framework only if the addition is made through plugpoints.

Q22. Elaborate on "REP" in detail.

Answer :

The Release/Reuse Equivalency Principle(REP) suggests that the classes must be packed into the packages such that they should be convenient for other users to reuse. The packages consisting of the collection of classes must be tracked and released.

The author of the class/package must take some care about the packages that he/she develop. The author must notify the reusers in advance about any changes in the packages that he/she wants to perform. The author has to maintain and support the older versions of the entities when the users are upgrading towards the newer version of the entities. This can be achieved by establishing a release contract system.

The REP states that the developer should not simply claim the reusability of the components but also provide the tracking system, notifications, support and safety to the potential reusers.

The REP not only focuses on reusability of the components but also it focuses on the reusers who use these components i.e., it should allow the same users to reuse all the classes of compatible component(package) but it should not allow a user to reuse the classes that are incompatible.

Q4. List the principles proposed for metrics characterization and validation.

Answer :

The following are the principles proposed for metric characterization and validation,

1. A mathematical measurement or a specified range must be assigned to every metric. For instance, if a metric has 0 to 1 range then here '0' indicates an absence of data point, '0.5' indicates a half-way point and '1' indicates the maximum value. In addition, the metric that attempts to be measured on a rational scale should not constitute components measured on an ordinal scale.
2. A software attribute defined by a metric increases when positive characteristics are encountered and decreases when negative characteristics are encountered. Thus, the value of the metric must increase/decrease in a uniform way.
3. Before a metric is used for making decisions or advertised, it should be practically validated in several different domains. Thus, the metric should be applicable to large systems, operate in several different system devices and programming languages and also be able to measure several aspects of interest irrespective of other aspects.

Q5. Define Testing.

Answer : (Model Paper-III, Q1(g) | Nov./Dec.-17(R13), Q1(g))

Testing

Testing is the process used to estimate the productivity and quality of software. It is developed by providing the necessary details about the software such as efficiency, portability, usability, capability, etc. Testing not only performs program execution but also detect bugs that halts the execution of a program. Testing is done by comparing the static and dynamic behavior of the software with the specifications described during the process of test design.

Testing Process

Testing refers to the process of determining errors and debugging defects in order to generate error free software. The testing process involves three phases.

1. Test planning
2. Test case design
3. Test case execution.

Q6. Why is a highly coupled module is difficult to unit testing?

Answer :

Unit testing focuses on testing small units or modules of a system. The purpose of testing is to be find as many errors in the system as possible.

"Highly coupled" modules are difficult to unit testing because the level of dependency between them is very high. Dependency among modules exists in several ways.

(i) One module refers to many number of modules.

(ii) A huge number of parameters are passed from one module to another module.

(iii) High level of complexity in the interface among modules.

(iv) One module has great amount of control over other modules. Because of the high level of inter connection between modules it is very difficult to break highly coupled modules into small units for testing to trace errors and to debug. Hence, we can say highly coupled modules are difficult to unit test.

Q7. Define unit testing and top-down integration testing.

Answer : Model Paper-I, Q1(h)

Unit Testing

The process of executing a single unit/module without effecting other modules present in the software and comparing actual outputs with the predefined outputs in the component-level design description document is called unit testing. It concentrates on testing the data structures and internal processing logic of a module. Moreover, various units can be simultaneously tested by parallely performing unit testing for each unit. There are two strong and supporting reasons for unit testing.

Top-down Integration Testing

When the integration starts from the main control module of main program and moves down in the control hierarchy. Then such integration is called "top-down integration". Moreover the down ward movement can be either depth-wise i.e., depth-first or breadth-wise i.e., breadth-first. For example in the below control hierarchy.

Q8. What are the advantages and disadvantages of bottom-up integration?

Answer :

Advantages of Buttom-up Integration

1. Stubs are eliminated as the processing needed for component subordinates is present all the time.
2. Logic modules are thoroughly tested.

Disadvantages o. Buttom-up Integration

1. Drivers are must for testing.
2. High-level modules are tested late in the testing process.
3. Early prototype is not possible.

Q9. What is regression testing? Give example.

Answer : Nov./Dec.-16(R13), Q1(h)

A well designed and well coded software when subjected to some changers may not work properly and require testing to ensure its proper working without any errors. This retesting of software is known as "Regression testing". For example, in case of integration testing, new modules are added or old modular are removed from the software which may change the I/O, control logic and data flow paths. Regression testing is then necessary to ensure that these changes are acceptable and do not result in any errors.

The most important task to be considered in regression testing is to select the tests in the test suite. The tests selected should be optimal and must uncover maximum errors from major parts of the program.

Moreover, the regression test suite must have the tests cases of,

(a) Modified software components,

(b) Software functions that may get affected due to modifications.

(c) All software functions.

Regression testing can be conducted either manually or by capture/playback tools. These tools help the software engineer to determine the tests that were conducted on the old system and also the results obtained. The old results are compared with newly generated results. The tests cases used in testing should be document properly such that these tests can also be implemented in future in case of further changes in software.

Regression testing is not limited to integration testing as it is also carried out while software maintenance process is in progress.

Q10. What is black box testing?

Answer :

Black box testing is also known as functional testing or behavioural testing. It is an effective and efficient approach used to concentrate on the inputs, outputs and principle function of a software system module. In a block-box testing, the test of a software is based on system specifications rather, than on code. Thus, the system is assumed to be 'black box' whose behavior can only be determined by studying its inputs, outputs and functional requirements of the software. The tests can be observed from the program codes or component specifications.

Q11. How is white box testing performed using statement testing?

Answer :

Test values are provided to check whether each statement in the module is executed at least once. Statement testing executes statements when,

(i) Optional arguments are available

(ii) User-provided parameters or procedures are available

(iii) Planned user-actions are available

(iv) Defined error codes are available.

Q12. Define configuration review and software tools.

Answer : Model Paper-II, Q1(h)

Configuration Review

Configuration review ensures that software configuration's elements are appropriately developed and recorded. Moreover, these elements should also contain sufficient information to the support phase in the software life cycle. Configuration review is the most important element of software validation. It is also known as an "audit".

Software Tools

Software tools for debugging give automated assistance in debugging the problems of software. The purpose of debugging tools to help the programmer find these problems quickly and efficiently. These tools are approached when manual debugging is difficult to implement and time consuming. Most of the available debugging tools are particular to environment and a programming language.

Q13. Define recovery testing.

Answer :

The most important and desirable feature of a computer-based system is "fault tolerance". Meaning, the system if effected by some faults or failures should resume processing within a stipulated period of time. Moreover, the system should be strong enough to overcome the failures and function normally. To known whether a system is fault-tolerant, try to fail the system in all aspects and measure its recovering/resuming capabilities. This type of system testing is called recovery testing.

The system under test may either use automatic recovery or human-intervened recovery approach. In the former case, the system itself performs recovery, but does require that the correctness of checkpoint mechanisms, reinitialization, restart and data recovery is checked. In the latter, manual recovery is done. MTTR (Mean Time to Repair) is the only factor used to check whether manual recovery is within the specified limits.

Q14. Define metrics.

Answer :

A metric is defined as a quantitative measure of having a certain attribute in a system/process/component i.e., the degree upto which the process may constitute a specified attribute. Typically, every software is dependent on every measure such as the average number of errors detected are dependent on the unit test/review of the system component.

A software metric can do the following,

1. Aid the design such that an efficient testing can be done.
2. Indicate that the source code and design are associated with the complexity measures.
3. Aid to derive the measures and analysis of design models.

Q15. Explain the merits and demerits of use-case-oriented metrics.

Answer :

Merits of Use-case-oriented Metrics

- Usecase can be used as a normalization measure similar to LOC or FP.
- Usecase provides information about the functions in the software.
- Usecase represent features of the applications.
- The count of usecases is proportional to,
  (a) The size of the software in LOC and
  (b) The count of test cases to be developed which is capable of testing the entire software.

Demerits of Use-case-oriented Metrics

- Since usecases are created at different levels of abstractions, they do not have any standard size.
- Without any standard measure the application of the usecase as a normalization measure is often suspected.

Q16. What guidelines should be applied when we collect software metrics?

Answer :

Guidelines for collecting software metrics,

1. Interpret metrics data carefully.
2. Do not ignore the metrics data that show a problem area because they are the way to improve the process.
3. The teams/individuals who collect measures and metrics should be given regular feedback.
4. Do not threaten teams/individuals using metrics.
5. Fix appropriate goals and metrics. This can be done by working with teams and practitioners.
6. Metrics should not be used to judge individuals.
7. Do not focus only on one metric isolating all other important metrics.

Q17. Discuss a testing strategy for object-oriented architectures.

Answer :

While dealing with software testing strategy for object oriented architecture, following are few significant changes required to be made to conventional testing.

1. Here, new testing techniques favouring the error discovery procedures should be launched in order to actively test various analysis as well as the design models.
2. The initial phase of testing i.e., unit testing possesses only little value in this aspect, also significant changes are needed to be made to integration testing.
3. While the object oriented representations are built their completeness and consistency should be assessed.

An object oriented software is tested initially for small parts of the code which are referred to as 'classes'. The testing is than carried out for the large software. The classes possesses attributes and operations and can implement communications and collaborations. After these classes are integrated to form a single Object-oriented architecture, regression testing is performed to eliminate side-effects due to introduction of new classes and to uncover collaboration and communication errors. The last testing step is testing the entire system in order to uncover requirement errors.

Q18. What are the metrics used for software maintenance?

Answer : Nov.-15(R13), Q1(h)

The following are the metrics used for software maintenance,

(a) **Direct Metrics**

Direct metrics are the software metrics that can be measured directly without using any functionality, complexity.

(b) **Indirect Metrics**

Indirect metrics are the metrics that are measured indirectly. When compared to direct metrics, indirect metrics are more difficult to collect.

(c) **Public Metrics**

Public metrics are software process metrics that usually consists of information, which were private to software project teams and to individual member of project team.

(d) **Private Metrics**

Private metrics are the software process metrics (data) that consists of information, which is private to individual software engineers.

Q19. List the metrics for design model.

Answer : (Model Paper-III, Q1(h) | Nov./Dec.-17(R15), Q1(h))

The metrics for design model are used to measure design attributes in such away that they aid the software engineers in evaluating the quality of design. Such metrics comprise of the following,

(i) **Metric of Interface Design**

This metric is basically focused on usability.

(ii) **Metric of Components**

This metric is used to evaluate the complexity of software architectural components as well as those aspects (characteristics) that are dependable on quality.

(iii) **Metric of Specialized Object-oriented Design**

This metric is used to measure classes, their communication aspects as well as the collaboration aspects.

(iv) **Metric of Software Architecture**

This metric is used to define the quality of software architectural design.

Q20. What is a software measure? Explain the importance of it in detail.

Answer :

Software Measure

Software measure is simply defined as the quantitative representation of the amount, dimension, extent, capacity or size of some attribute belonging to a process or a product. A measure is established when a single data point (Example, the uncovered errors of single software component) is identified.

Importance of Software Measure

A software measure is important for,

- Estimating and developing the metrics which are useful in obtaining the indicators (metric or combination of metrics through which the software process, project or product can be obtained).
- Assessing the quality of the product.
- Obtaining the effectiveness in test cases.
- Building the overall quality of the software.
- Measuring out the plan.
- Forecasting the relationship between models product and process.
- Improving the quality of the product.
- Enhancing the performance of the process.

UNIT 5

RISK AND QUALITY MANAGEMENT

SIA GROUP

PART-A
SHORT QUESTIONS WITH SOLUTIONS

Q1. Define risk management.

Answer :

Risk management is a process of identifying and managing the potential risks that might affect the project schedule or the quality of a software. The software tools for risk management are used to assist the project team in order to identify, access and minimize the risks from the software project.

Q2. Explain about product size risks.

Answer :

Model Paper-I, Q1(I)

Product size risks are the risks associated with the overall size of the software that is to be developed or modified. The probability for the occurrence of product size risk is 30% and its impact on the organization is castastrophic that results in significant degradation of technical performance.

The following are certain risk item issues considered with respect to product size risks,

1. What is the estimated product size as well as the level of confidence with the estimated size?
2. What is the size of database allocated to product for both creating and using it?
3. What is the total count of the users using the product?
4. What are the modifications to be performed on the requirements of a product?

Q3. Distinguish between generic risks and product specific risks.

Answer :

The difference between generic risks and product specific risks are,

	Generic Risks		Product Specific Risks
1.	Generic risks are essential threat to every software project.	1.	Product specific risks are essential threat that is specific to software project.
2.	Generic risks can be identified easily.	2.	Product specific risks can be identified by examining of project plan and software scope.
3.	Generic risks occurs due to uncertainties involved in accessing or estimating various inputs to the software process.	3.	Product specific risks occurs due to conditions and constraints about resources, customers and lack of organization support.
4.	Generic risks are simple to understand.	4.	Product specific risks are difficult to understand.

Q4. List the steps in risk projection.

Answer :

Following are the four risks projection steps,

1. Develop a measure using which probability of existence of risk can be known.
2. Depict the outcome of a risk.
3. Project the effects which can be observed on both the project as well as on the end product being developed.
4. Finally, measure the appropriateness (i.e., trueness) of the risk projection.

Q5. Write a brief note on risk assessment.

Answer : Model Paper-II, Q1(i)

Risk Assessment

Whenever risks becomes inevitable, following are the three factors through which their (risks) effects on current environment can be determined.

1. **Risks Nature**

   This factor describes the problems associated with the given risk.

2. **Risks Scope**

   This factor describes the intensity (how deeply the given risk affects) of the risk in the current situation.

3. **Risks Timing**

   This factor describes the duration during which the given risk will exert its effect.

Q6. Define risk refinement.

Answer : (Model Paper-III, Q1(j) | Nov./Dec.-17(R15), Q1(i))

Risk refinement is a process of refining the already stated risk definition and representing it in a more detailed form. The process of refinement is initiated when project planning process is thoroughly analyzed. This refinement has been proved useful to mitigate, monitor and manage these risks in a more convenient manner. The process can be carried out by following a standard notation of risk representation referred to as Condition Transition Consequence (CTC). This notation states that, "If a condition is specified, then there exist a concern that may possibly result in a consequence".

Q7. Define quality of conformance.

Answer :

Quality of conformance refers to an extent that the software being manufactured satisfies all the specifications issued by the designers. Quality of conformance looks only for implementation of quality design characteristic issues as its target objective. Hence, to have good quality of conformance one has to see that the production of software must satisfy all design specifications, requirement specifications and also quality specifications which, were laid prior to the software development process.

One of the latest approximations delivered by a scientist Robert Glass suggests software quality to be one of the entities of user satisfaction.

User satisfactor = A complaint product + High quality + Delivery of product within the estimated schedule and time

Q8. Discuss the importance of quality assurance.

Answer : Model Paper-III, Q1(i)

Quality Assurance (QA) is the activity where certain auditing and reporting functions are performed in order to assess the completeness and effectiveness of quality control activities. The quality assurance is important because,

(i) It helps in defining ways in which software quality can be achieved.

(ii) It helps development organizations to examine whether the required quality level has been achieved or not.

(iii) It helps to reduce the amount of rework that will result to lower costs.

(iv) QA professionals look software from customer's point of view and hence act as their representatives. As a result the desired product is built keeping in mind the customer's ease.

Q9. Discuss briefly about process standards.

Answer :

They refer to standards that are applied while the software is being developed. It specifies the process of designing, implementing and validating the software product. These two standards are important for the following reasons,

(i) The standards are built by selecting the most appropriate or best practice that a company follows. These best practices may be result of long research and experience of the company. By following standards one can avoid repeating mistakes made in past.

(ii) By following standards, all engineers will adopt the same set of activities or practices. As a result, work done by one person can be easily understood and continued by others.

(iii) If standards contain best practices, then the framework provided by them can be used to implement quality assurance. Hence, Quality Assurance (QA) is important for both customer and development organization to evaluate the quality of a given software product.

Q10. Explain the relationship between quality and security of software.

Answer :

The importance of security of a software is increasing along with the growth in criticality of web-based system and applications. A low quality software can easily be hacked. As a result the security risk of it increases with its problems and costs.

Basically, there are two types of problems that occur in software,

1. Occurrence of bugs that are the problems occurred during the implementation.
2. Flaws in software due to improper architectural plans.

The security completely relates to the quality. People aware of the above problems should focus on early phases of the life cycle stuff rather than in later phases.

The bugs are paid much attention than the flaws elimination of architectural flaws makes the software hacking difficult. Therefore, people should focus on quality at design phase itself in order to build a secure system.

Q11. When will be the formal technical reviews are conducted? And what are its objectives?

Answer :

FTR is conducted when software engineers wants to review a product at any phase of the development process. It involves software engineer and a review team.

The main objectives of FTR are as follows,

(a) To check whether software meets the requirements.

(b) To check the uniformity of the software process.

(c) To verify that software representation is as per the predefined standards.

(d) To discover errors of the software with respect to its operation, logic and implementation.

(e) To make the projects in a controllable way.

Q12. What is the importance of software reviews.

Nov./Dec.-16(R13), Q1(j)

Answer :

The importance of software review are as follows,

(i) The correction of errors that are found initially in the process is cost-effective.

(ii) Errors are amplified along with progression of the process. Hence, the minor errors that are not corrected initially can be amplified into a big set of errors later in the project.

(iii) Reviews reduce the amount of rework that is required late in the project. By this, lot of time is saved.

Q13. Define software reliability. (Nov./Dec.-17(R15), Q1(j) | Nov./Dec.-16(R13), Q1(l))

OR

What is software reliability? Define. (Model Paper-I, Q1(j) | Nov.-15(R13), Q1(l))

Answer :

Software reliability refers to the probability of software running with out any failure. The software failures mainly occur due to the design pattern followed or due to implementation problems but the software doesn't wear out like the hardware components. Software availability refers to probability that a program or software is available.

Q14. Can a program be correct and still not exhibit good quality? Explain. (Model Paper-II, Q1(j) | Nov.-15(R13), Q1(j))

Answer :

Yes, it is possible that in spite of correct program, it may not exhibit good quality.

A program may support the quality of conformance i.e., confirming to all functional and performance requirements writhing the scheduled time. Degree of quality of conformance is high when the resulting system meet the requirements and performance criteria.

If a program is designed accurately ad generates correct output, still there is possibility of hidden errors which causes the failure of a system. A system fails due to the following hidden errors,

(i) Poor quality control

(ii) Inadequate architectural design

(iii) Performance specifications like expected volume of the data, traffic congestion, transactions are insufficient in nature.

1. **Requirements Identification**

   Each of the requirements must be identified individually so that they can be referred by other requirement later and also to be used in traceability assessments.

2. **Change Management Process**

   For answer refer Unit-II, Q54.

3. **Traceability Policies**

   The traceability policies show the relationship that exists between the requirements and the system design and requirements which has to be recorded and maintained.

4. **Case Tool Support**

   These are the tools (ranging from specialist requirements management system to spreadsheets to simple database systems) that are used to process large amounts of information about the requirements. There are many relationships that exist between the requirements and system design. These relationships correspond to the links between the requirements and the reasons behind the proposal of these requirements. The impact of the changes on other requirements and system design must be traced when they are proposed.

   For remaining answer refer Unit-II, Q52.

2.3 SYSTEM MODELS

2.3.1 Context Models

Q56. What is meant by context model? With a neat diagram explain the context model of ATM system.

Model Paper-II, Q5(b)

OR

Elaborate on context models.

Answer :

Context Model

Nov./Dec.-12(R09), Q3(b)

In the process of software development we should decide on the system boundary and its environment at an early stage in the requirements elicitation and analysis process. Once we are done with this a simple architectural model is brought up. For example following is an architectural structure representing the context of Bank ATM system.

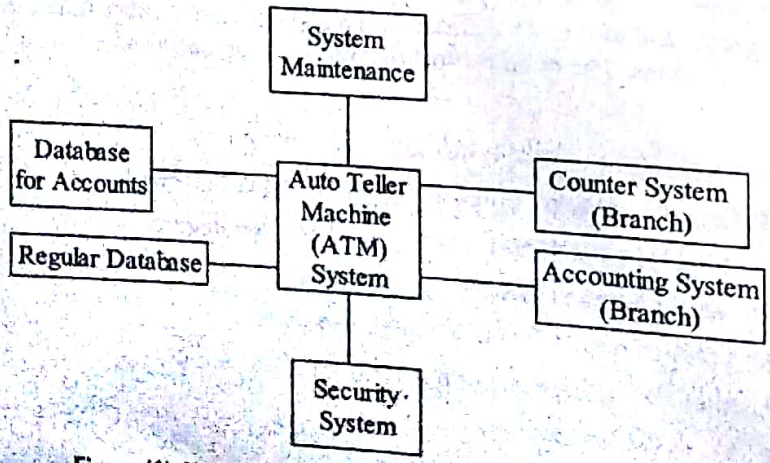


Figure (1): High Level Representation of Bank ATM System

The figure (1) represents only the block diagram which is nothing but a high level representation of the ATM system. In figure (1) the rectangle represents specific subsystem and embedding the name of the subsystem within it. The lines connecting these rectangles represent the collaborations existing between these subsystem. However, the figure (1) depicts only the high level view of the system context. It does not express any details of various other independent systems working with it. Other external systems should be considered since they may produce or intake the data produced to or from the current system, their presence may have drastic impact on the current system, they may be directly connected to the current system etc.

Hence, process model should be augmented with the given architectural model proposed. The process model includes all the activities supported by the system. Following model is nothing but the example process model of equipment procurement.

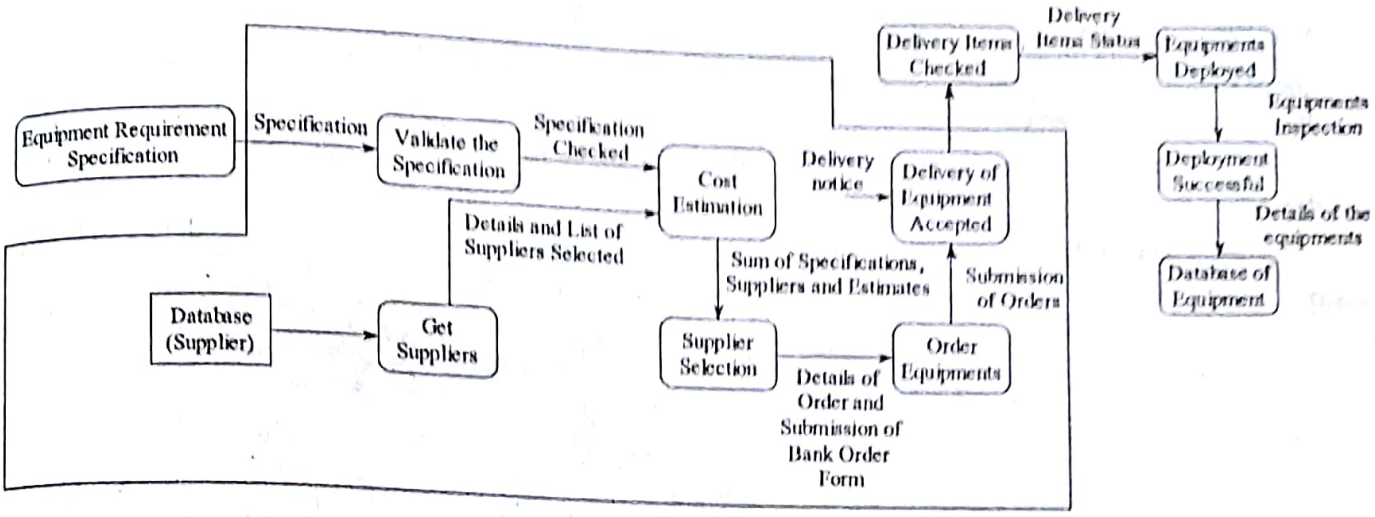


Figure (2): Process Model of Equipment Procurement

In figure (2), certain activities lie inside the system boundary, (as shown by the box) while the other activities lie outside the system boundary. Hence, when the computer support has been specified for the process model, certain decisions regarding the activities that are helpful must be taken. The activities of the process model include the following. They are,

(i) The equipments that are essential need to be specified.

(ii) The suppliers need to be determined and selected.

(iii) The equipments should be ordered.

(iv) The ordered equipments need to be delivered.

(v) Finally, testing should be performed to check whether the equipments have been delivered or not.

2.3.2 Behavioral Models

Q57. What is a behavioural model? Discuss various types of behavioural models with examples for each.

Answer :

Behavioural Model

Behavioural model is a process to illustrate the entire functionality of the system. It provides the dynamic view of the system, where the changes in development of a software process are done dynamically. It is used to describe the behavioural activities of the system based on the states, events and time variant. Behavioural model is also defined as a function of specific events and time, which indicates how software will respond to these external events and state of changes during software development.

These are two kinds of behavioural models in software development such as,

1. Data flow model
2. State machine model.

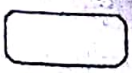
1. Data Flow Model

Data flow model depicts processing of data at each stage of system development. The model reflects the procedure used in data processing when considered along with analysis prospective.

Notations Used in Data Flow Diagram

We usually use three types of symbols in each data flow diagram. These symbols and their description are given below,

 – Rectangles are used to represent the data stores.

 – Rounded rectangles are used to depict the functional processing in the system.

 – Arrows represent flow of data from one to another function.

Following are certain important facts which are effective while working with data flow model,

- ❖ Data flow model depicts the implementation of data processing actions at each stage of system development.
- ❖ After data is suitably processed it is passed to next stage of processing.
- ❖ The processing which is depicted at each stage, is nothing but the real time functions or processing need to be accomplished in real time software development.
- ❖ Data flow diagrams are often represented so that the analysts can gain information on the scenario being described by these models.
- ❖ These are rather most simple and easy while developing.

Example

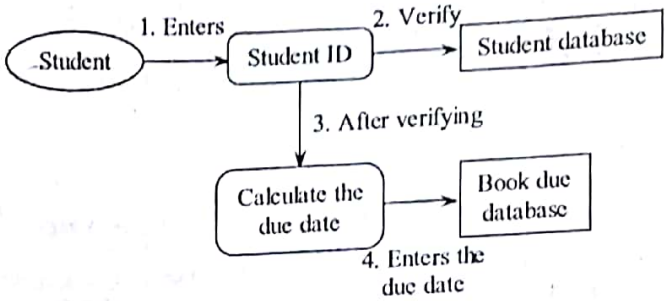


Figure (1): DFD of Library System

If a student wants to borrow books from the library then the following steps are performed,

(i) First, the student enters the student ID.

(ii) The student ID is verified by searching the details in student database.

(iii) If the verification is done successfully, then the due date on which the student must return the borrowed book is calculated.

(iv) The calculated due date value is entered into the books due database.

2. State Machine Model

The state machine model is used to model the real time behaviour of a given system. It shows the states and events that result in change of the system's current states. However, it does not show how the data flow within the system. Hence, state machine model consists of states, events (external/internal) and transitions between these states.

State machine model assumes that, at a given instance, the process may remain at any state and whenever it receives an event it proceeds to the next state. For example, consider a computerized pressure controlling system in a reactor, the pressure in the reactor may be lowered from current pressure value when an operator performs an operation. Hence, lowering of pressure is nothing but an example of event generation.

Example

Consider a simple microwave oven. It contains several buttons to,

(i) Set power

(ii) Set time

(iii) Start the system and

(iv) Cancel the operation.

The steps required to cook food in an microwave oven are,

(i) Set the level of power to be consumed it can be either full or half.

(ii) Set the time required for the food to be cooked.

(iii) Press the start button to allow food to be cooked.

The state machine model of a microwave oven is shown below. It contains rectangular boxes that depicts states and arrows represents state transitions. Each state contains a small description within it as 'do' statement.

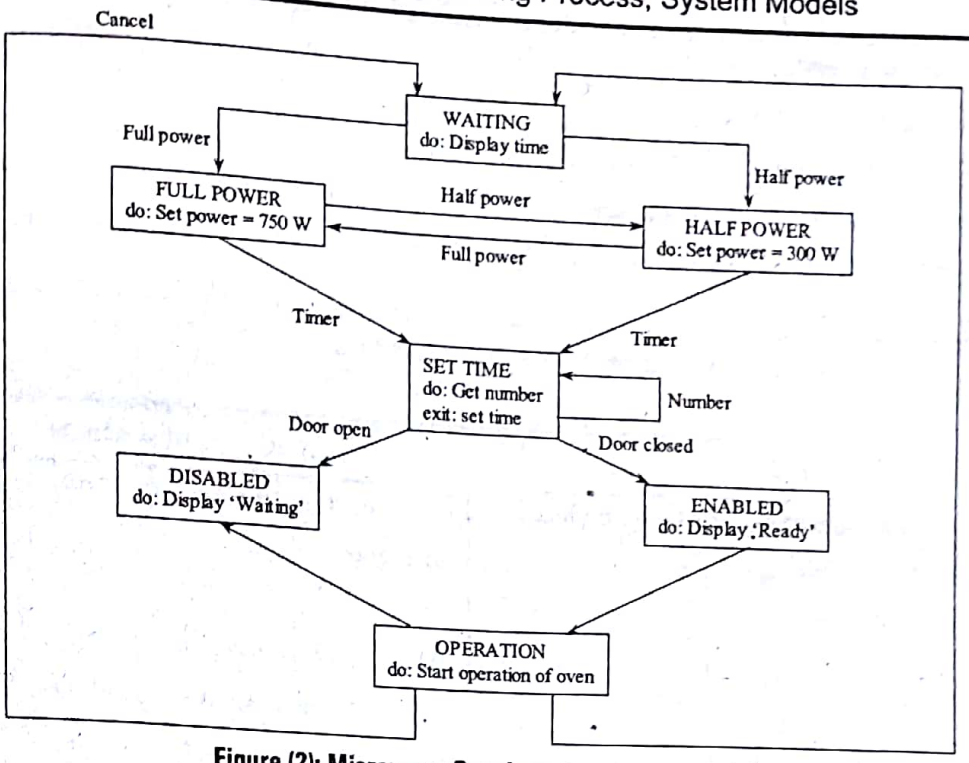


Figure (2): Microwave Oven's State Machine Model

Initially, the microwave oven will be in 'waiting' state. To use the microwave oven, firstly, the level of power to be consumed is set. It can be set either to 'full power' or 'half power'. The 'waiting' state will then be changed either to the full power or half power state. User is allowed to change the power from full to half power and half to full power at any time. After the power is set, the time needed to cook food is set in 'set time' state. Then, the door of the microwave should be closed. If it is not closed, then state will change from 'set time' state to 'disabled'' state and if it is closed, then it will changed from 'set time' state to 'enabled' state. When microwave oven is in 'enabled' state, the start button can be pressed to start its operation. When this operation is carried out, it can either be cancelled or let to complete. In both the cases, the state of the microwave oven will change from 'operation' state to 'waiting' state.

Q58. What is data flow diagram? Draw a data flow diagram for order processing system.

Answer :

Dec.-11, Set-3, Q6(a)

Data Flow Diagram

For answer refer Unit-II, Q57, Topic: Data Flow Model (Exclude Topic: Example).

Example

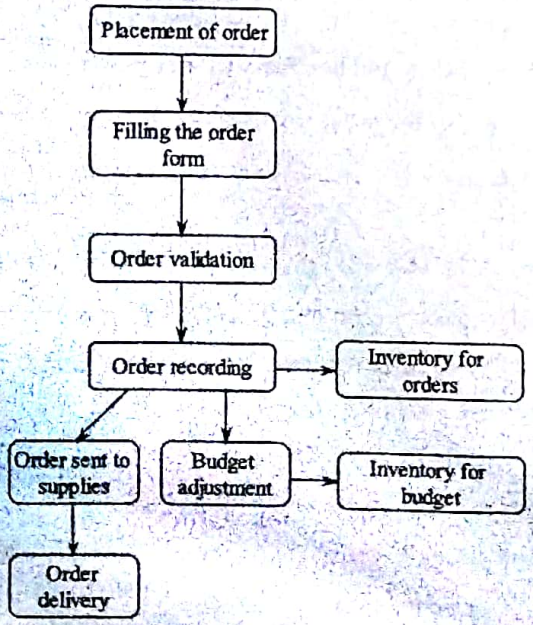


Figure: Order Processing System

Q61. With an example, explain the data models. Also state few advantages of using the data dictionary.

Answer :

Model Paper-I, Q5(b)

Data Models

Database forms one of the essential aspects of any of the systems being developed (irrespective of their sizes). Also we have models to represent the processing of logical data belonging to these systems and such models are often referred as semantic data models. Most of the software development organizations today rely on entity relation attribute modelling for modelling database systems. The main aspects of these models are the entities, attributes and their relationships that exists among these entities. This gained wider recognitions even in object oriented database modelling.

A new language referred as unified modelling language has acquired the position of entity relation attribute modelling. UML directly did not support the database modelling, but has spreaded its roots in encouraging models for semantic data modelling. UML adopted objects, classes strategies in each of its models. Classes are represented as the entities of era model.

The following example depicts semantic data model using UML class notations,

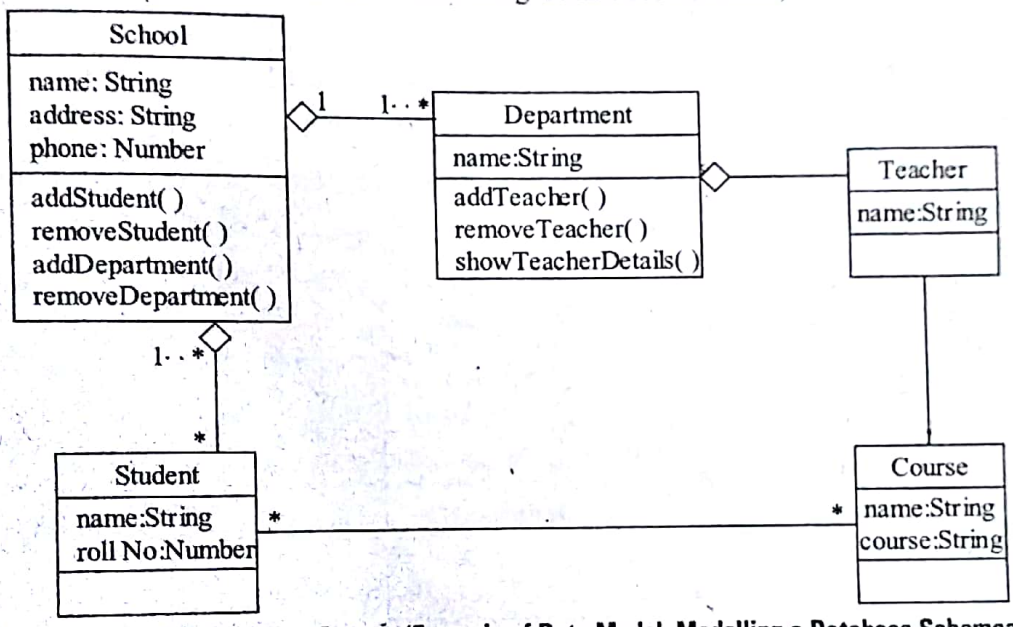


Figure: School Information System (Example of Data Model, Modelling a Database Schemea)

On thorough analysis of the above model, we can conclude that, the model does expose most of its details. As details are often a necessary aspect, they need to be stored in specified storage locations in an orderly format. Such storage locations are often referred as data dictionaries and all the names are stored in the these dictionaries in an alphabetical format. Apart from names, the dictionary can also maintain a variety of other information associated with these names. There are immense advantages of maintaining data dictionaries.

Advantages of Data Models

These advantages are stated below,

- **It can be Applied as a Major Tool for Name Management**

  While developing large enterprise software, usually the developers may introduce new name for a given entity. The data dictionary software ensures that, no two similar names exists because of which no confusion occurs.

- **It can be Used for Storing Large Information at Organization and Level**

  Hence, using data dictionary, all the related information can be stored at one place causing an ease while adding, retrieving, analyzing the important information.

For example, if the information about the library system is to be stored in the data dictionary, then its probable field can be,

Name	Description	Type	Date
Authors	Description of entities	Attribute	17.07.07

Figure: Data Dictionary

2.3.4 Object Models

Q62. What is object model? Explain various types of object models with examples.

Answer : Model Paper-III, Q5(b)

Object Model

Generation of object models are carried out for the system incorporating the object-oriented programming methodology.

Basically, object models are used to develop interactive systems. This means, these systems are being implemented by using object oriented programming language like C++ or JAVA.

During requirement analysis, object models that are developed can represent both system data and its processing.

Object models are natural representation of real world entities called object classes, which have their own attributes and operations. These object classes are required in information processing system.

For instance entities may be student, book, account etc. The attributes of a student entity may be name, roll no, student id etc.

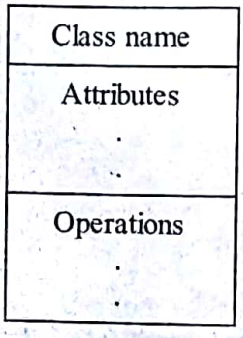
An object class consists of three sections,

(i) Class name

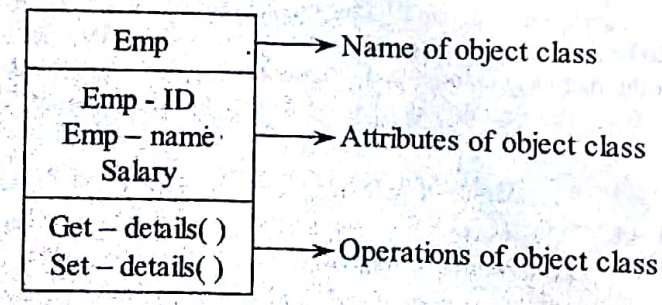
(ii) Attributes of class

(iii) Operations of class.

In UML, it is graphically represented by a rectangle as,



Example



All these things can be represented by using object model.

In order to simplify the system design, object model is combined with data flow model.

Different types of object models can be created depending on the activities of object i.e., object aggregation, object interaction etc.

Types of Object Models

There are three different types of object models. They are,

1. Inheritance model
2. Object aggregation
3. Object behaviour model.

1. Inheritance Model

In object modeling, a group of classes are identified and if there exists some common attributes and services then inheritance property is implemented.

Inheritance reflects the property of deriving one class property into other classes. The property of the class being derived is referred as parent class and the classes acquiring these properties are referred to as child classes. Consider a simple diagram depicting the animal kingdom.

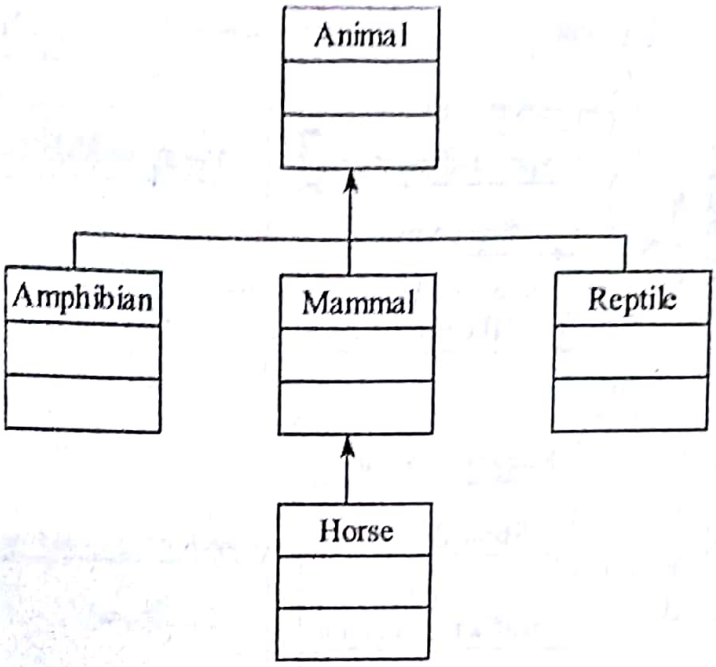


Figure: An Example of Inheritance Hierarchy

In the figure inheritance is shown by upward arrow.

Here, 'Animal' is a base class through which three child classes are inherited namely, 'amphibian', 'mammal' and 'reptile. Here, further 'mammal' acts as a base class through which one child class is inherited namely, 'Horse'.

2. Object Aggregation

Sometimes we may encounter certain situations where, a single object may have multiple objects associated with it. To model such situation we usually use object aggregation model. Following is an example, depicting the modelling of various components of a computer system.

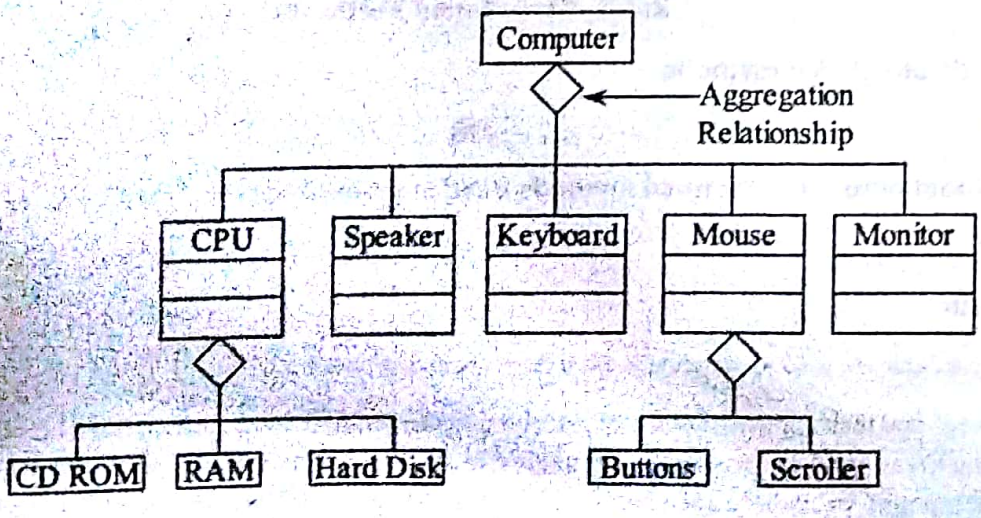


Figure: Depicting Aggregation Relationship Existing Among Various Objects

3. Objects Behaviour Model

In object behaviour modelling, the behaviour of objects is represented by the sequence of actions take place. In the UML, this is modeled using the sequence diagram.

In a sequence diagram, actors and the objects are placed at the top of the diagram and labelled arrows represent the operations of system. The vertical rectangle specifies the time span of the particular operation. The following is an example of sequence diagram that shows the sequence of actions generated during ATM withdrawal transaction.

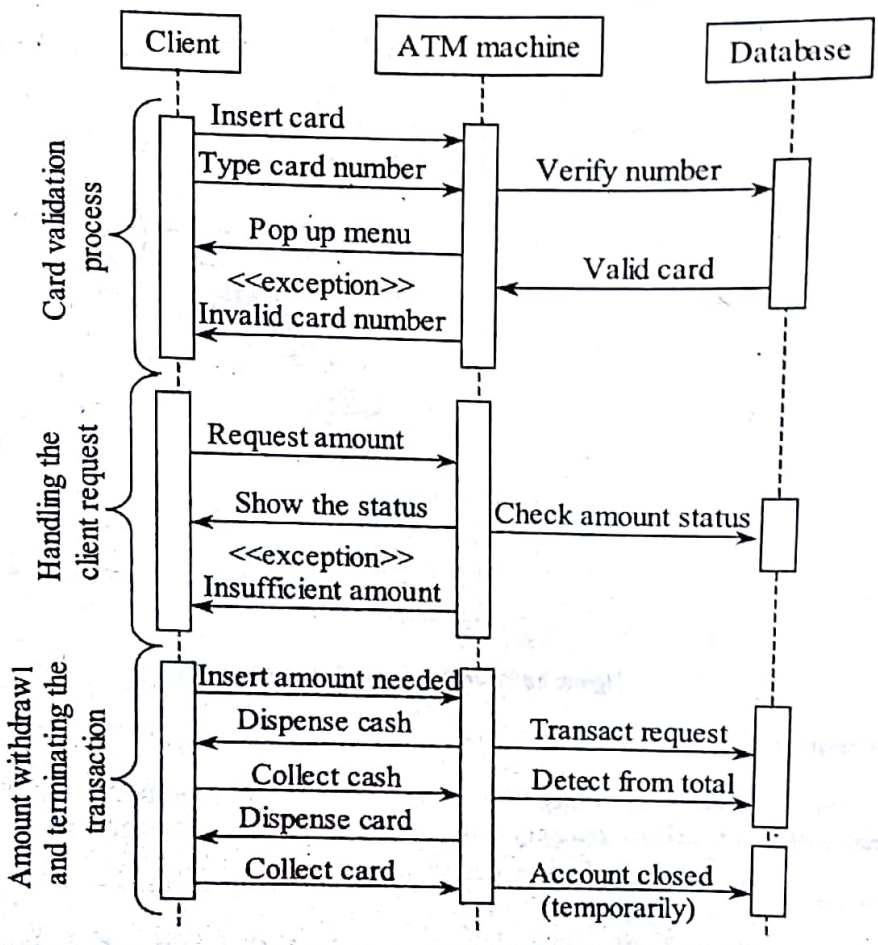


Figure: Sequence Diagram Depicting ATM Withdrawl Transaction

UNIT 4

TESTING STRATEGIES, PRODUCT METRICS, METRICS FOR PROCESS AND PRODUCTS

Marketed by SIA GROUP

PART-A
SHORT QUESTIONS WITH SOLUTIONS

Q1. What is meant by software testing? Also, define verification and validation.

Answer :

Software Testing

Model Paper-I, Q1(g)

Software Testing is a method in which the process of executing a given software is executed with an intention of detecting bugs in it. It is one of the essential steps in software development life cycle. Hence, it is the responsibility of the programmer to build an error free software. The characteristics of software testing are,

Verfication and Validation

"Verification" and "validation" are two important aspects of the software testing process, though these two phrases seems to have similar definitions, but there exists huge difference between them – verification is the process of ensuring that the given software satisfies almost all the credentials which were laid prior to it's (software's) development and validation is the process of ensuring that the given software satisfies the customer requirements.

Q2. Differentiate between verification and validation.

Answer :

Nov./Dec.-16(R13), Q1(g)

"Verification" and "validation" are two important aspects of the software testing process, though these two phrases seems to have similar definitions, but there exists huge difference between them – verification is the process of ensuring that the given software satisfies almost all the credentials which were laid prior to it's (software's) development and validation is the process of ensuring that the given software satisfies the customer requirements. Hence, in short we can refer both of them as,

Verification

It checks whether a product is being built in the right way.

Validation

It checks whether the right product is being developed.

Hence, in order to ensure that the given software is correct in all aspects, then it has to satisfy the verification and validation process successfully.

Q3. Discuss the statement, "Build robust software that is designed to test itself".

Answer :

Model Paper-II, Q1(g)

Build a robust software that is designed to test itself mean a software design should be established in such a way that it withstands in all the worse conditions or in case of failure. A software design should exhibit all the characteristics of high quality. A software should detect all the bugs. And to do this it should use antibugging techniques that easily detects the errors and resolves them. The software design should allow regression testing and automated testing.