MCA(4th Sem)

Software Engineering (Set-I)

Q. 1) Answer the following questions **(2x10)**

a) Mention the importance of phase containment errors. Relate phase containment errors with the development cost of software.

Ans: - Importance of Phase Containment Errors Phase containment of errors means detect and correct errors as soon as possible. It is an important Software Engineering Principle. A software development life cycle has different distinct development phases. Phase containment of errors means detect and correct the errors within the phase where it's actually lives. That is a design error should be detected and corrected within the design phase itself rather than detecting it in the coding phase. To achieve phase containment of errors we have to take periodic reviews.

Phase containment is used for defect prevention so that the defects are found at each phase of the application, and are not multiplied further; we do it by taking JAD sessions, performing reviews after each phase, by participating in walkthroughs and inspection.

b) Justify the following:
"Spiral lifecycle model can be considered to be a Meta model".

Ans: - Spiral Model is called Meta model because it is composed of several other models. For example, a single loop spiral actually represents the Waterfall Model. The spiral model uses prototyping approach before embarking on the actual product development effort. Also the spiral model can be considered as supporting the evolutionary model-the iterations along the spiral can be considered as evolutionary levels through which the complete system is built. This enables the developer to understand and resolve the risks at each evolutionary level. The spiral models uses prototyping as a risk reduction mechanism and also retain the systematic step approach of waterfall model.

c) Compute the nominal effort and development time for *m* organic type software product with an estimated size of 500,000 line of code.

Ans: - Effort=$a_b(\text{KLOC})^{1.05}$ Development time=$c_b\ (\text{KLOC})^{0.38}$
Given, lines of code = 500 KLOC

$$E=2.4(500)^{1.05} = 1637.30$$
$$D=2.5(500)^{.38} =26.51$$

d) List any two factors that affect the productivity of a software development team.

Ans:- Two Factors that Affect the Productivity of a Software Development Team
Technical knowledge in the area of the project (domain knowledge) is an important factor determining the productivity of an individual for a particular project, and the quality of the product that he develops. A programmer having a thorough knowledge of database applications (e.g., MIS' may turn out to be a poor data communication engineer. Lack of familiarity with the application with the application areas can results in low productivity and poor quality of the product. Since software development is a group activity, it is vital for a software engineer to possess three main kinds of **communication skills**; Oral, Written, and interpersonal.

e) Mention any **two** causes that may lead a high coupling between two software modules.

Ans:- Causes of High Coupling
Coupling increases between two classes A and B if:
A has an attribute that refers to (is of type) B.
A call on services of an object B.
A has a method which references B (via return type or parameter).
A is a subclass of (or implements) class B.

f) Write the main differences between structured chart and flow chart' in the context of software design.

Ans: - A structure chart differs from a flow chat in following

i. It is usually difficult to identify different modules of the software from its flow chart representation.
ii. Data interchange among different modules is not represented in a flowchart.
iii. Sequential ordering of tasks inherent in a flowchart is suppressed in a structure chart.
iv. A structure chart has no decision boxes.
v. Unlike flow charts, structure charts show how different modules within program interact and data that is passed between them.

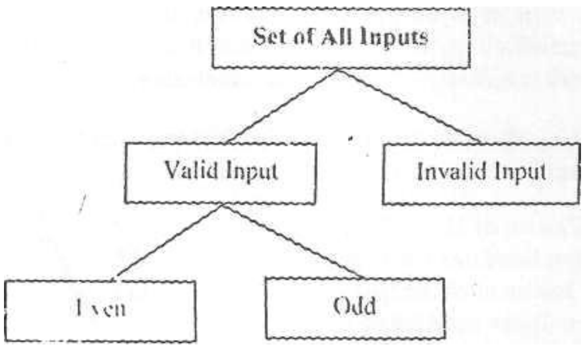
g) Compare the relative advantages of code inspection and code walk through.

Ans: Advantages of Code Inspection and Code Walk Through: - **Code inspection and code walk through helps to avoid the faults.**

- Lists all potential design flows that can make s/w code less maintainable and costly to develop.
- Indicates all defects otherwise difficult to testing and usage.
- A detailed error feedback' is provided to individual programmers. So that it make easier for them to make changes in the code.
- Error reduction and cost reduction.
- Improvement in productivity.

h) Design equivalence class partitioning test suite that reads an integer value and display whether it is even.

Ans: The equivalence classes are even numbers, odd ad invalid inputs. Selecting one representative value from each equivalence class.



i) Name any four metrics to ensure software reliability.

Ans: Four Metrics to Measure Software Reliability are:-

- Rate of Occurrence of Failure (ROCGF)
- Mean Time To Failure (MTTF)
- Mean Time to Repair (MTTR)
- Mean Time between Failures (MTBF)

j) Write any two advantages of total quality management.

Ans: Advantages of Total Quality Management

- Elimination of non-confirmation and repetitive work
- Elimination of waste costs and reject products
- Elimination of repairs and reworks
- Process efficiency leading to improved profit per product or service

Q. 2)

a) Mention the important phases of spiral model. Compare the relative merits and dements of spiral model and the prototype model of software development.

Ans: Spiral Model

The spiral' model (proposed by Boehm) is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the linear sequential model.

The Spiral Life Cycle Model combines elements of the waterfall lifecycle model, along with an emphasis on the use of risk management techniques.

Boehm proposed a spiral model where each round of the spiral:

- Identifies the sub-problem which has the highest risk associated with it and
- Finds a solution for that problem.

Its chief distinguishing feature is that it is primarily risk-avoidance driven rather than document-driven, code driven or release driven.

It is best suited for highly risky projects as the spiral model uses prototyping as a risk reduction mechanism while following the systematic, sequential waterfall approach.

The phases of Spiral model are explained below.

Planning: The objectives, limitations and alternatives of the project are found and documented in the planning phase. The strategies to follow during the lifecycle are decided .in this phase.

Risk Analysis**:** The very critical part of Spiral methodology is Risk analysis phase. The available and possible ways of doing cost effective development of project are decided in this phase. The possible risk that may occur during the development is studied in this phase. If any risk factors are found, the ways to eliminate them are also planned at this stage itself.

- Engineering: The practical development of project takes place in this step. The outcome in this stage is passed iteratively through all phases to obtain modifications or improvement in the project.
- Evaluation: Customer's evaluation of the project is done at this phase. The customers can give their suggestions and ideas to identify and solve problems in the final software. The phase is similar to testing phase in waterfall model.

Merits and Demerits of Prototyping Model

Strength	**Weakness**	**Types of Projects**
Prototyping Helps in requirements elicitation Reduces risk and leads to a better system.	Front heavy process Possibly higher cost Disallows later changes.	System with novice users When uncertainties in requirements.
Spiral Controls project risks very flexible Less documentation needed.	No strict standards for software development. No particular beginning or end of particular phase.	Projects built on Untested assumptions.

b) Write the different types of risks that software might suffer from. Explain three essential activities for risk management.

Ans: Types of Risks
Project Risks: Project risks threaten the project plan, that is, if project risks become real, it is likely that project schedule will slip and that costs will increase. Project risk: identify potential budgetary, schedule, personnel (staffing and organization), resource, customer, and requirements problems and their impact on a software project. Project complexity, size, and the degree of structural uncertainty were also defined as project (and estimation) risk factors.

Technical Risks: Technical risks threatened the quality and timeliness of the software to be produced. If a technical risk becomes a reality, implementation may become difficult or impossible. Technical risks identify potential design, 'implementation, and interface, verification, and maintenance problems. In addition, specification ambiguity, technical uncertainty, technical obsolescence, and a leading edge technology are also risk factors. Technical risks occur because the problem is harder to solve than we thought it would be.

Business Risks: Business risks threaten the viability of the software to be built. Business risks often jeopardize the project or the product. The top five business risks are:

- Building an excellent product or system that no one really wants (market risk).
- Building a product that no longer fits into the overall business strategy for the company (strategic risk),
- Building a product that the sales store doesn't understand how to sell
- Losing the support of senior management due to a change in focus or a change in people (management risk), Losing budgetary or personnel commitment (budget risks).

Risk Management Activities

Major risk management activities include identify, analyze, plan, track and control risks. These activities serve as the foundation for the application of continuous risk management.

Each risk normally goes through these activities sequentially.
Step 1: Risk Identification Recognizing what can go wrong is the first step, called "risk identification"

Step 2: Risk Analysis: Next each risk is identified to determine the likelihood that it will occur and the damage that it will do it if does occur.

Step 3: Risk Prioritization: Once risk analysis information is established, risks are ranked, by probability and impact.

Step 4: Risk Control: Finally, a plan is developed to manage those risks with high probability and high impact.

Q.3)

a) If a software product for business application costs Rs. 1, 00,000 to buy and that its size is 60 KLOC assuming that in house developers cost Rs, 10,000 per month.

Ans:- Form the basic COCOMO estimation. Formula for organic software:

Effort = 2.4 x (60)105 = 73.63PM

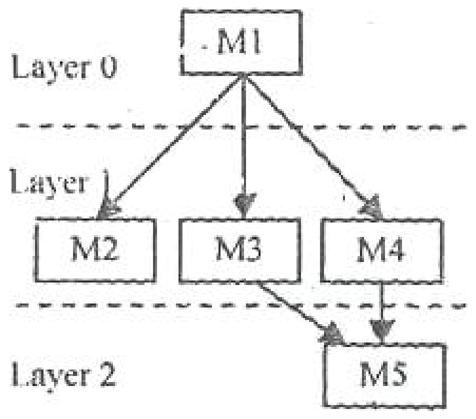
Nominal development time= 2.5 x $(73.63)^{0.38}$ = 12.8062 months

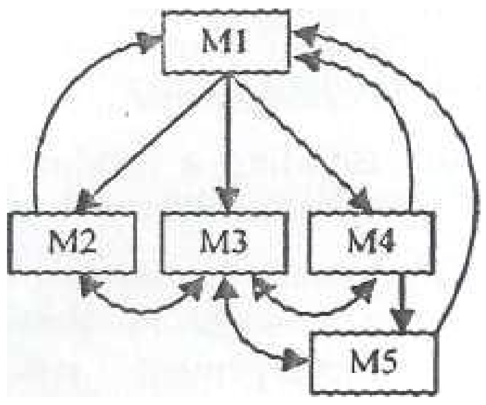
Cost required to develop the product = 12 x 10,000= 1, 20,000

So, more cost-effective is to buy the product.

b) Discuss the important concepts associated with a layered design.

Ans:-An important characteristic feature of a good design solution is layering of the modules. A layered design achieves control abstraction and is easier to understand and debugs.





a) Layered design with good control abstraction b) Layered design with poor control abstraction

In a lavered design. the topmost module in the hierarchy can be considered as a manager that onlv invokes the services of the lower level module to discharge its responsibilitv. The modules at the intermediate lavers offer services to their higher laver bv invoking the services of the lower laver modules and also bv doing some work themselves to a limited extent. The modules at the lowest laver are the worker modules. These do not invoke services of any module and entirely carry out their responsibilities by themselves.

Understanding a lavered design is easier since to understand one module one would have to at best. consider the modules at the lower lavers (i.e.. the modules whose services it invokes). Besides. in a lavered design errors are isolated. since an error in one module can affect onlv the higher laver modules. As a result. in case of anv failure of a module. onlv the modules at the lower levels need to be investigated for the possible error. Thus, debugging time reduces significantly m a layered design.

On the other hand. if the different modules call each other arbitrarilv. then this situation would correspond to modules ar; -aged in a single laver. Locating an error would be both difficult and time-consuming. This is because. once a failure is observed. the cause of failure (i.e.. error) can potentiallv be in anv module. and all modules would to be investigated for the error. In the following, we discus; some important concepts and terminologies associated with a layered design.

Q. 4)

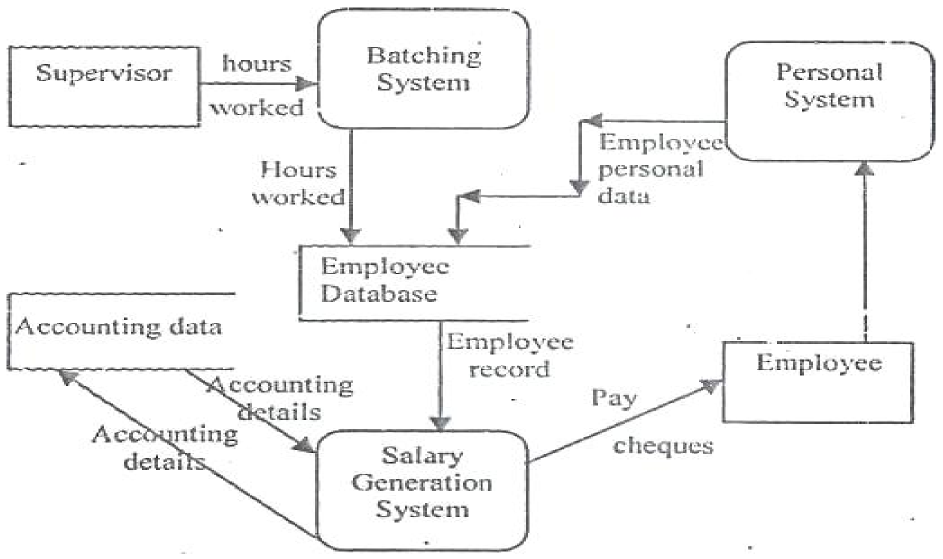
a) Compare the relative advantages and disadvantages of function-oriented design and object-oriented design.

Ans:- Function-Oriented Design versus Obiect-Oriented Design come of the differences between the functional oriented and the obiect-oriented approaches which are very indispensable are described as under in the tabular form:

Functional-Oriented Approach	Object-Oriented **Approach**
In functional-oriented design approach, the basic abstractions which are given to the user are real- world functions such as sort, merge, track, display, etc.	In object-oriented design approach, the basic abstractions are not the real-world functions, but the data abstraction where the real-world entities are represented such as picture, machine, radar system, customer, student, , employee, etc.
In function-oriented design, functions are grouped together by which a higher-level function is obtained. For example, SA/SD.	In this, the functions are grouped together on the basis of data they operate on like in class person, function display are made member functions to operate on its data members like the person name, age, etc.
In this approach, the state' information is often represented in a centralized shared memory.	In this approach, the state information is not represented in a centralized shared memory, out is implemented/distributed among the objects of the system.

b) Draw the Data flow diagram of Payroll management software.

Ans: - Data flow diagram of Payroll management software:-



Q. 5)

a) Explain the important steps to be followed during the architectural design of distributed systems.

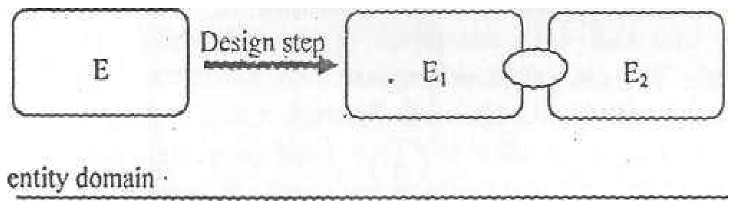
Ans:- : In a top-down design, the design of the distributed system is manipulated in successive design steps, which make it possible for designers to move from an abstraction level to a more concrete one. Each design step brings the resulting design closer to the implementation, in terms of designs that can be more easily mapped onto concrete components.

An entity may represent at a certain abstraction level, a complex structure of more concrete entities. An (interaction may also represent, at a certain abstraction level, a complex structure of more concrete interactions. Therefore, during the design trajectory of distributed systems one can identify the following design steps:

Entitv Decomposition: Aoolied to a set of entities in a design, it replaces souse or each of these entities bv multiole entities.
Interface Refinement: Aoolied to a set of (inter)actions in a design, it replaces some or each of these (interactions by multiple (interactions.

Since the entitv and behavior domains are related, i.e., an entitv should have a behavior assigned to it: entitv decomoosition also affects the behavior domain. This means that the original behavior of an entitv that is decomoosed should also be decomoosed so that each resulting entitv has a corresoonding behavior assigned to it. Methods are necessary to guarantee the correctness of design operations in which behaviors are decomposed.

The Figure given below shows a simole examole, consisting of an entity E that is decomposed in two entities Ei and E_2, which are connected via an interaction point.



In this examole, both actions a and b are executed bv entitv E, such that a is followed bv b. In the decomoosed design, action a is assigned to E_1 and action b is assigned to E_2. Intuitivelv one can deduce that an interaction between Ei and E_2, which we call c, is necessarv in order to keeo the original relationshio between actions a and b. Unfortunatelv, not all instances of behavior refinement are so simple as this one in which correctness can be easily and intuitively assessed by hand.

An examole of interface refinement is the decomoosition of interaction c of figure 3 in two successive interactions c_i and c_2, reoresenting an indication that Bi is readv to oass some control information to B_2 and a notification that B_2 has accepted this information from Bj.

b) Define Real time system. Mention the essential activities in design of Real time systems.

Ans: - Real Time Svstem
Real time svstem is one that must react to inouts and resoonds to them quicklv. A real time svstem has well defined, fixed time constraints. The design and administration of tests and the selection of test equioment: Real-time svstem is anv svstem that resoonds m a timelv manner. Activities must be scheduled and executed to meet their timeliness requirements. Timeliness requirements (and svstems) are often classified into hard real-time, where failure to meet a deadline is treated as catastrophic system failure: and soft real-time, where an occasional missed deadline may be tolerated.

Characteristics of Real Time Systems

- Timeliness: Time is a resource of fundamental concern in real-time svstems.
- Embedded Svstems: Thev are comoonents of a larger svstem that interacts with the ohvsical world. This is often the orimarv source of comolexitv in real-time svstems. The ohvsical world tvoicallv behaves in a non- deterministic manner, with events occurring asvnchronouslv, concurrentlv, and in an unoredictable order. Whatever haooens, the embedded real-time system must respond appropriately and in a timely fashion.
- Concurrencv: The concurrencv of the ohvsical world usuallv imolies that real-time svstem software must also be concurrent. In fact, one of the orimarv requirements of manv real-time svstems is to svnchronize multiole concurrent activities arising in the environment. Unfortunatelv, concurrencv adds comolexitv to software design since it conflicts with the inherentlv serial, cause-and-effect flow of human reasoning.
- Deoendabilitv: Another salient characteristic of manv real-time svstems is deoendabilitv. Manv real-time svstems olav a crucial role in their environments and failure to oerform correctlv mav result in significant costs or risk human safetv. As a result, real-time svstems must often be highlv reliable (i.e., they must perform correctly), and available (i.e., they must operate continuously).

Essential Activities in Design of Real Time Systems

1. Selection of the hardware and software and the appropriate mix for a cost-effective solution;
2. Decision to use a commercial RTOS or-to design a special OS (or dedicated application without an OS);
3. The selection of software language;
4. Maximizing fault tolerance and reliability through careful design and rigorous testing;
5. The design and administration of tests and the selection of test equipment;

Q. 6)

a) Write essential steps to test a system through white box testing. Write how white box testing is different from black box testing.

Ans:- Steps to Test a System through White Box Testing White box testing involves the following steps:
1. Create test plans. Identify all white box test scenarios and prioritize them.
2. Profile the application block. This involves studying the code at run time to understand the resource utilization, time spent by various methods and operations, areas in code that are not accessed, and so on.
3. Test the internal subroutines. This step ensures that the subroutines or the nonpublic interfaces can handle all types of data appropriately.
4. Test **loops** and conditional statements. This step focuses on testing the loops and conditional statements for accuracy and efficiency for different data inputs.
5. Perform security testing. White box security testing helps you understand possible security loopholes by looking at the way the code handles security.

White-Box Testing versus Black-Box Testing

1. White-box testing is testing on more detailed level at the earlier stage of software development, it covers testing of the code itself (i.e. unit testing). .Programming knowledge is usually required for white-box testing.
2. White-box testing is also sometimes called clear-box testing. The name comes from the fact that the inner part of the software (code) is tested so you get a view inside the software.
3. Black-box testing is testing of software functionality. Programming skills are usually not required. This is testing more from the end-user point of view. It is called black-box testing because during the testing you don't get to see the inner structure (i.e. code) of the software.

b) Mention the importance of program analysis tools in software development Explain the different types of program analysis tools.

Ans: Importance of Program Analysis Tools:-A programming tool or software development tool is a program or application that software developers use to create, 'debug, maintain, or otherwise support other programs and applications. The term usually refers to relatively simple programs that can be combined together to accomplish a task, much as one might use multiple hand tools to fix a physical object.

Programming tool or programming software is a sub- category of system software but sometimes it is stated as a separate category of software along with application and system software

Program analysis tools can be classified in to two broad categories
1. Static **Analysis Tools:** Static analysis tools assess and compute the various characteristics of a software product without executing, it.Typically static analysis tools analyze some structural representation of a program to arrive at certain analytical conclusion, e.g. that some structural properties hold. Static analysis tools often summarize the result of analysis of every function in a polar chart known as Kiviat chart. A kividt chart typically shows the analyzed values for cyclomatic complexity, number of source line, percentage of comment lines, Halstead's metric, eU..
2. Dynamic Analysis Tools: Dynamic program analysis techniques require the program to be executed and its actual behavior recorded. A dynamic analyzer usually instruments the code. The instrumented code when executed allows us to record the behavior of the software for different test cases. After the software has been tested with its full test suite and its behavior recorded, the dynamic analysis tool carries out a post execution analysis

and produce reports which described the structural coverage that has been achieved by the complete test suit for the program.

Normally the dynamic analysis results are reported in the form of a **histogram** or pie **chart** to describe the structural coverage achieved for different modules of the program.

Q. 7)

a) Draw the control flow graph for the function find largest, which will find the largest number from a set of input numbers. From the control graph determine its cyclometric complexity.

Ans: - The control flow graph to find maximum from N number of inputs is shown in the figure given below.

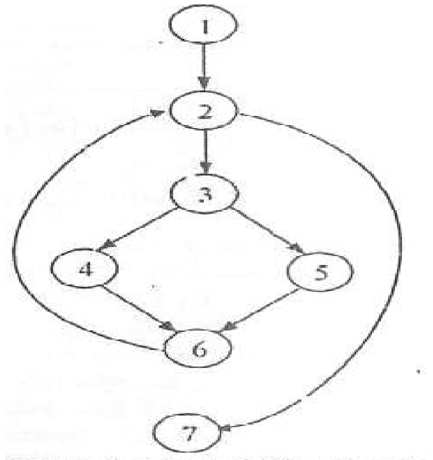


Figure 4: Control Flow Graph

Cyclomatic Complexity:-

Cyclomatic complexity is defined for each graph to be e - n + 2, where e and n are the number of edges and nodes in the control flow graph, respectively. Cyclomatic complexity is also known as v(G), where v refers to the cyclomatic number in graph theory and G indicates,that the complexity is a function of the graph.

Cyclomatic Complexity for Maximum number of n inputs

From the figure, the number of edges e =8 Number of nodes n. =7

Hence cyclomatic complexity V (G) = 8 - 7 +2 = 3

b) Write the. importance of statistical testing. Mention the important steps associated with it. **Ans:** Statistical Testing

Statistical testing is a testing process whose objective is to determine the reliability of the product rather than discovering errors. The test cases designed for statistical testing have an. entirely different objective from that of conventional testing. To carry-out statistical testing, we need to first define the operation profile of the product;

1. Operation Profile: Different categories of users may use software for very different purposes. I or example, a librarian might use the Library Automation Software to create member records. delete member records, and add books to the library, etc., whereas a library member might use the same software to query about the availability of a book, or to issue and return books. Formally, we can define the operation profile of software as the probability distribution of the input of an average user. If we divide the input into a number of classes {Ci}, the probability value of a class represents the probability of an average user selecting his next input from this class. Thus, the operation profile assigns a probability value {Pi} to each input class {Ci}.

2. How to Define the Operation Profile for a Product? We need to divide the input data

into a number of input classes. For example, for the graphical editor software, we might divide the input into data associated with the edit, print, and file operations. We then need to assign a probability value to each input class: this probability value would signify the probability for an input value from that class to be selected. The operation profile of a software product can be determined by observing and analyzing the usage pattern of a number of users.

3. Steps in Statistical Testing: The first step is to determine the operation profile of the software. The next step is to generate a set of test data corresponding to the determined operation profile. The third step is to apply, the test cases to the software and record the time between each failure. After a statistically significant number of failures have been observed, the reliability can be computed. For accurate results, statistical testing requires some fundamental assumptions to be satisfied. It requires a statistically significant number of test cases to be used. It further requires that a small percentage of test inputs that are likely to cause system failure be included. It is straightforward to generate test cases for the common types of inputs; it is easy to write a test case generator program which can automatically generate these test cases. But, it is also required that a statistically significant percentage of the unlikely inputs should also be included in the test suite. Creating these unlikely inputs using a test case generator is very difficult.
4. Pros and cons of Statistical Testing: Statistical testing allows one to concentrate on testing parts of the system that are most likely to be used. Therefore, it results in a system that the users find more reliable (than actually it is!). Also, the reliability estimation arrived at by using statistical testing is more accurate compared to those of other methods discussed. However, it is not easy to do the statistical testing properly. There is no simple and repeatable way of defining operation profiles. Also, the number of test cases with which the system is to be tested should be statistically significant.

Q. 8) Write short notes on any Two of the following: (5x2)

a) Project estimation technique.

b) Coupling and cohesion

c) System testing

a) Project estimation technique.

Estimation of various project parameters is a basic project planning activity. The important project parameters that are estimated include: project size, effort required to develop the software, project duration, and cost. These estimates not only help in quoting the project cost to the customer, but are also useful in resource planning and scheduling. There are three broad categories of estimation techniques:

1. Empirical Estimation Techniques: Empirical estimation techniques are based on making an educated guess of the project parameter: While using this technique, prior experience with development of similar products is helpful. Although empirical estimation techniques are based on common sense, different activities involved in estimation have been formalized over the years.
2. Heuristic Techniques: Heuristic techniques assume that the relationships among the different project parameters can be modeled using suitable mathematical expressions. Once the basic (independent) parameters are known, the other (dependent) parameters can be easily determined by substituting the value of the basic parameters in the mathematical expression.
3. Analytical Estimation Techniques: Analytical estimation techniques derive the required results starting with basic assumption regarding the project. Thus, unlike empirical and heuristic techniques, analytical techniques do have scientific basis. Halstead's software science is an example of an analytical technique. Halstead's software science can be used to derive some interesting results starting with a few simple assumptions.

b) Coupling and Cohesion:-

Coupling between two modules is a measure of the degree of interdependence or interaction between the two modules. A module having high cohesion and low coupling is said to be functionally independent of other modules. If two modules interchange large amounts of data, then they are highly interdependent. The degree of coupling between two modules depends on their interface complexity.

The interface complexity is basically determined by the number of types of parameters that are interchanged while invoking the functions of the module.

Classification of Coupling
Even if there are no techniques to precisely and quantitatively estimate the coupling between two modules, classification of the different types of coupling will help to quantitatively estimate the degree of coupling between two modules, rive types of coupling can occur between any two modules.

- Data coupling
- Stamp coupling
- Control coupling'
- Common coupling
- Content coupling

Cohesion
Most researchers and engineers agree that a good software design implies clean decomposition of the problem into modules, and the neat arrangement of these modules in a hierarchy. The primary characteristics of neat module decomposition are high cohesion and low coupling. Cohesion is a measure of functional strength of a module. A module having high cohesion and low coupling is, said to be functionally independent of other modules. By the term functional independence, we mean that a cohesive module performs a single task or function. A functionally independent module has minimal interaction with other modules

Classification of Cohesion
The different classes of cohesion that a module may possess are:

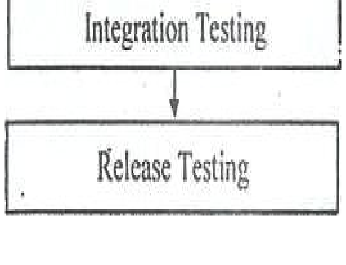
- Coincidental cohesion
- Logical cohesion
- Communications cohesion
- Functional cohesion

c) System Testing

System testing is a series of different tests whose primary purpose is to fully exercise the computer-based system. Bach test has a different purpose for the various system elements that have been properly integrated.

System testing involves integrating two or more components that implement system functions or features and then testing this integrated system is an iterative development process, system testing is concerned with testing an increment to be delivered to the customer; in a waterfall process, system testing is concerned with testing the entire system.

Phases in System Testing
For most complex systems, there are two distinct phases to system testing:



1. Integration Testing: Where the test .teams have access to the source code of the system. When a problem is discovered, the integration team tries to fmd the source of the problem and identifies the components that have to be debugged. Integration testing is mostly concerned with finding defects in the system.
2. Release Testing: Where a version of the system that could be released to users is tested. Hence, the test team is concerned with validating that the system meets its requirements and with ensuring that the system is dependable. Release testing is usually black-box testing where the test team is simply concerned with demonstrating that the system does or does not work properly. Problems are reported to the development team whose job is to debug the program. Where customers are involved in release testing, this is sometimes called acceptance testing. If the release is good enough, the customer may then accept it for use.