UNIT-I

- **Software Development Life Cycle Models**

- **Phases of Software Project**

Software life cycle models describe phases of the software cycle and the order in which those phases are executed. Each phase produces deliverables required by the next phase in the life cycle.

Requirements are translated into design. Code is produced according to the design which is called development phase. After coding and development the testing verifies the deliverable of the implementation phase against requirements.

There are following six phases in every Software development life cycle model:

- ✓ Requirement gathering and analysis
- ✓ Design
- ✓ Implementation or coding
- ✓ Testing
- ✓ Deployment
- ✓ Maintenance

- ✓ **Requirement gathering and analysis**

This phase is the main focus of the project managers and stake holders.

Meetings with managers, stake holders and users are held in order to determine the requirements like; Who is going to use the system? How will they use the system? What data should be input into the system? What data should be output by the system? These are general questions that get answered during a requirements gathering phase.

After requirement gathering these requirements are analyzed for their validity and the possibility of incorporating the requirements in the system to be development is also studied.

Finally, a Requirement Specification document is created which serves the purpose of guideline for the next phase of the model.

- ✓ **Design**

In this phase the system and software design is prepared from the requirement specifications which were studied in the first phase.

System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture. The system design specifications serve as input for the next phase of the model.

✓ **Implementation / Coding**

Since, in this phase the code is produced so it is the main focus for the developer. This is the longest phase of the software development life cycle.

✓ **Testing**

After the code is developed it is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase. During this phase unit testing, integration testing, system testing, acceptance testing are done.

✓ **Deployment**

After successful testing the product is delivered / deployed to the customer for their use.

✓ **Maintenance**

Once when the customers starts using the developed system then the actual problems comes up and needs to be solved from time to time. This process where the care is taken for the developed product is known as maintenance.

❖ **Quality**

It is brought about by strict and consistent commitment to certain standards that achieve uniformity of a product in order to satisfy specific customer or user requirements. ISO 8402-1986 standard defines quality as "the totality of features and characteristics of a product or service that bears its ability to satisfy stated or implied needs.".

❖ **Quality assurance (QA)**

Refers to administrative and procedural activities implemented in a quality system so that requirements and goals for a product, service or activity will be fulfilled.

Two principles included in QA are: "Fit for purpose", the product should be suitable for the intended purpose; and "Right first time", mistakes should be eliminated.

❖ **Quality control**

A process by which entities review the quality of all factors involved in production. This approach places an emphasis on three aspects:

✓ Elements such as controls, job management, defined and well managed processes,[1][2] performance and integrity criteria, and identification of records

- ✓ Competence, such as knowledge, skills, experience, and qualifications
- ✓ Soft elements, such as personnel, integrity, confidence, organizational culture, motivation, team spirit, and quality relationships.

The quality of the outputs is at risk if any of these three aspects is deficient in any way.

Quality control emphasizes testing of products to uncover defects and reporting to management who make the decision to allow or deny product release, whereas quality assurance attempts to improve and stabilize production (and associated processes) to avoid, or at least minimize, issues which led to the defect(s) in the first place.

Difference Between Quality Assurance and Quality Control

	Quality Assurance	**Quality Control**
Definition:	QA is a set of activities for ensuring quality in the processes by which products are developed.	QC is a set of activities for ensuring quality in products. The activities focus on identifying defects in the actual products produced.
Focus on:	QA aims to prevent defects with a focus on the process used to make the product. It is a proactive quality process.	QC aims to identify (and correct) defects in the finished product. Quality control, therefore, is a reactive process.
Goal:	The goal of QA is to improve development and test processes so that defects do not arise when the product is being developed.	The goal of QC is to identify defects after a product is developed and before it's released.
How:	Establish a good quality management system and the assessment of its adequacy. Periodic conformance audits of the operations of the system.	Finding & eliminating sources of quality problems through tools & equipment so that customer's requirements are continually met.
What:	Prevention of quality problems through planned and systematic activities including documentation.	The activities or techniques used to achieve and maintain the product quality, process and service.
Responsibility:	Everyone on the team involved in developing the product is responsible for quality assurance.	Quality control is usually the responsibility of a specific team that tests the product for defects.
Example:	Verification is an example of QA	Validation/Software Testing is an example of QC

	Quality Assurance	Quality Control
Statistical Techniques:	Statistical Tools & Techniques can be applied in both QA & QC. When they are applied to processes (process inputs & operational parameters), they are called Statistical Process Control (SPC); & it becomes the part of QA.	When statistical tools & techniques are applied to finished products (process outputs), they are called as Statistical Quality Control (SQC) & comes under QC.
As a tool:	QA is a managerial tool	QC is a corrective tool

❖ **Testing**

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Also to assess the feature of A software item. Testing assesses the quality of the product.

Software testing is a process that should be done during the development process. In other words software testing is a verification and validation process.

❖ **Verification**

Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to.

❖ **Validation**

Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements.

❖ **Process Model to Represent Different Phases**

A process model is a way to represent any given phase of software development that effectively builds in the concepts of validation and verification to prevent and minimize the delay between defect injection and defect detection (and eventual correction).

In this model, each phase of a software project is characterized by the following.

- ✓ Entry criteria, which specify when that phase can be started. Also included are the inputs for the phase.
- ✓ Tasks, or steps that need to be carried out in that phase, along with measurements that characterize the tasks.
- ✓ Verification, which specifies methods of checking that the tasks have been carried out correctly.

- ✓ Exit criteria, which stipulate the conditions under which one can consider the phase as done. Also included are the outputs for only the phase.

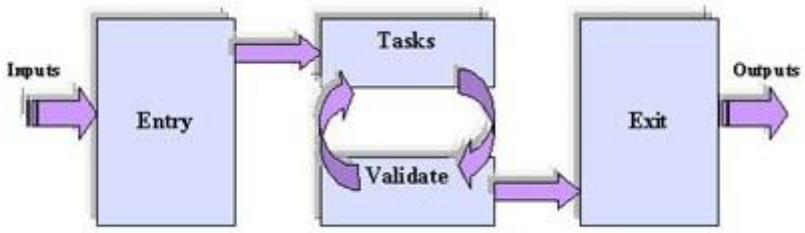


Figure-ETVX model applied to design

❖ Life Cycle Models

The software life cycle is a general model of the software development process, including all the activities and work products required to develop a software system.

A software life cycle model is a particular abstraction representing a software life cycle.

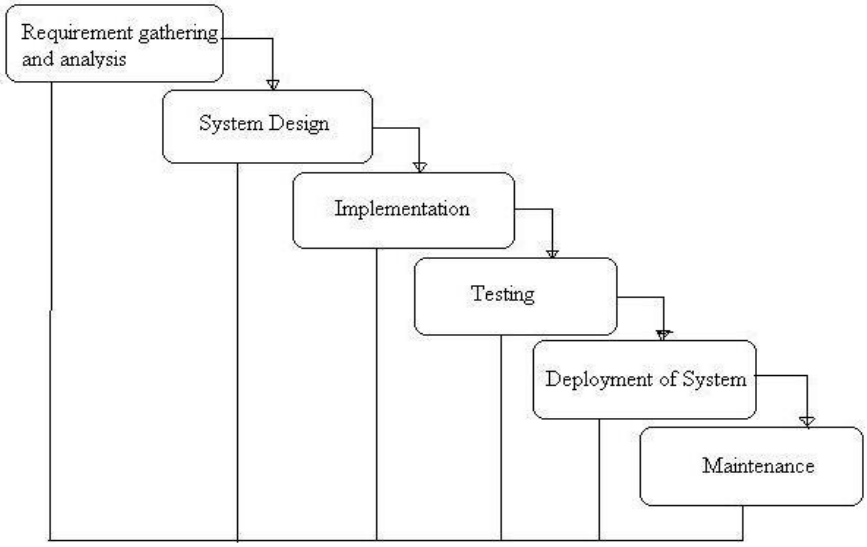
✓ Waterfall Model

The Waterfall Model was first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use.

In a waterfall model, each phase must be completed fully before the next phase can begin. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project. In waterfall model phases do not overlap.

Diagram of Waterfall-model:

General Overview of "Waterfall Model"



Advantages of waterfall model:

- ✓ Simple and easy to understand and use.
- ✓ Easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- ✓ Phases are processed and completed one at a time.
- ✓ Works well for smaller projects where requirements are very well understood.

Disadvantages of waterfall model:

- ✓ Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
- ✓ No working software is produced until late during the life cycle.
- ✓ High amounts of risk and uncertainty.
- ✓ Not a good model for complex and object-oriented projects.
- ✓ Poor model for long and ongoing projects.
- ✓ Not suitable for the projects where requirements are at a moderate to high risk of changing.

When to use the waterfall model:

- ✓ Requirements are very well known, clear and fixed.
- ✓ Product definition is stable.
- ✓ Technology is understood.
- ✓ There are no ambiguous requirements
- ✓ Ample resources with required expertise are available freely

- ✓ The project is short.
- ✓ **Prototyping Model**

The Prototyping Model is a systems development method (SDM) in which a prototype (an early approximation of a final system or product) is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can now be developed.

This model works best in scenarios where not all of the project requirements are known in detail ahead of time. It is an iterative, trial-and-error process that takes place between the developers and the users.

There are several steps in the Prototyping Model:

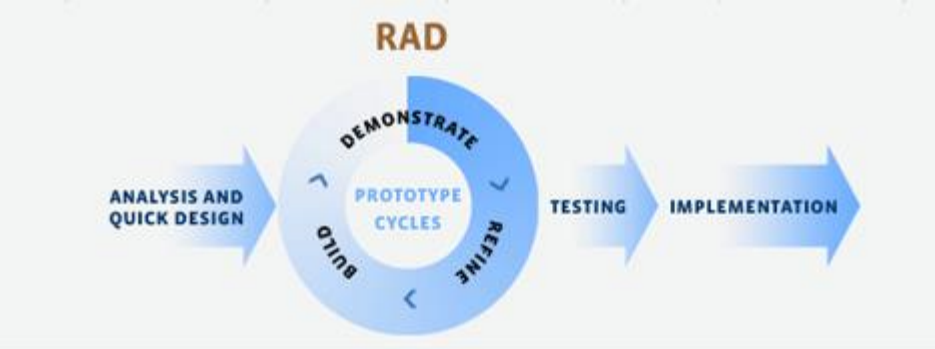
- ✓ The new system requirements are defined in as much detail as possible. This usually involves interviewing a number of users representing all the departments or aspects of the existing system.
- ✓ A preliminary design is created for the new system.
- ✓ A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
- ✓ The users thoroughly evaluate the first prototype, noting its strengths and weaknesses, what needs to be added, and what should to be removed. The developer collects and analyzes the remarks from the users.
- ✓ The first prototype is modified, based on the comments supplied by the users, and a second prototype of the new system is constructed.
- ✓ The second prototype is evaluated in the same manner as was the first prototype.
- ✓ The preceding steps are iterated as many times as necessary, until the users are satisfied that the prototype represents the final product desired.
- ✓ The final system is constructed, based on the final prototype.
- ✓ The final system is thoroughly evaluated and tested. Routine maintenance is carried out on a continuing basis to prevent large-scale failures and to minimize downtime.

- ✓ **Rapid Application Development model**

In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype.

This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.

Diagram of RAD-Model:



The phases in the rapid application development (RAD) model are:

Data modeling: Information gathered from business modeling is used to define data objects that are needed for the business.

Process modeling: Data objects defined in data modeling are converted to achieve the business information flow to achieve some specific business objective. Description are identified and created for CRUD of data objects.

Testing and turnover: Test new components and all the interfaces.

Advantages of the RAD model:

- ✓ Reduced development time.
- ✓ Increases reusability of components
- ✓ Quick initial reviews occur
- ✓ Encourages customer feedback
- ✓ Integration from very beginning solves a lot of integration issues.

Disadvantages of RAD model:

- ✓ Depends on strong team and individual performances for identifying business requirements.
- ✓ Only system that can be modularized can be built using RAD
- ✓ Requires highly skilled developers/designers.
- ✓ High dependency on modeling skills
- ✓ Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

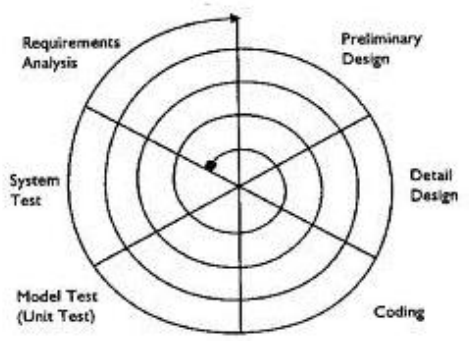
When to use RAD model:

- ✓ RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.
- ✓ It should be used if there's high availability of designers for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.
- ✓ RAD SDLC model should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time (2-3 months).

- ✓ **Spiral model**

The spiral model is a risk-driven process model generator for software projects.

Based on the unique risk patterns of a given project, the spiral model guides a team to adopt elements of one or more process models, such as incremental, waterfall, or evolutionary prototyping.



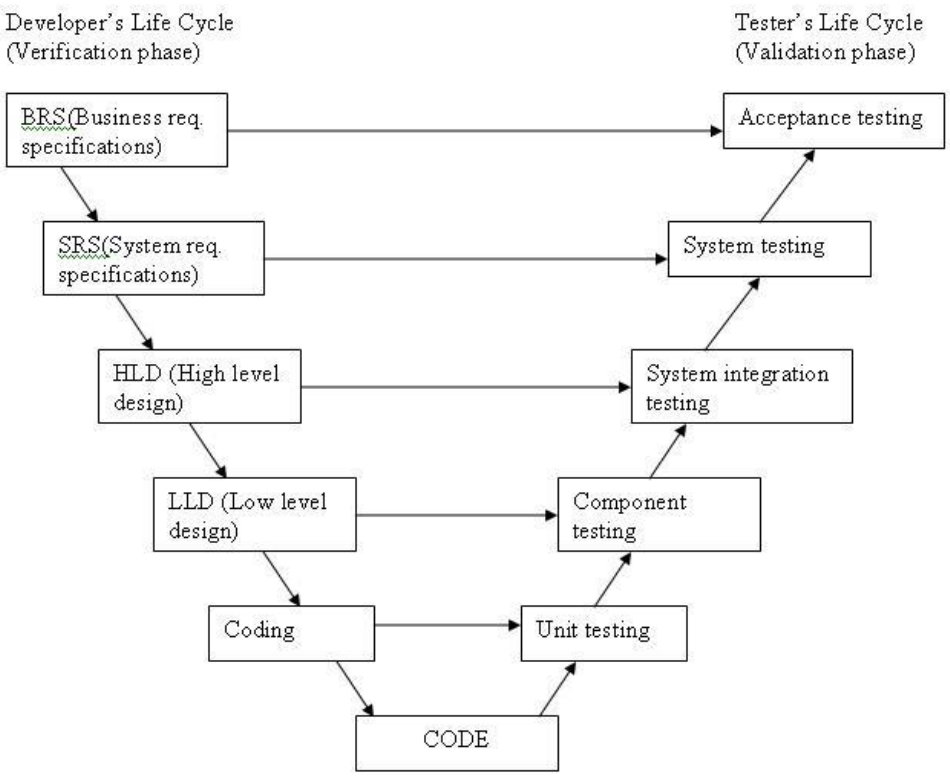
Spiral Model

- ✓ **The V Model**

V- model means Verification and Validation model. Just like the waterfall model, the V-Shaped life cycle is a sequential path of execution of processes.

Each phase must be completed before the next phase begins. Testing of the product is planned in parallel with a corresponding phase of development.

Diagram of V-model:



The various phases of the V-model are as follows:

Requirements like BRS and SRS begin the life cycle model just like the waterfall model. But, in this model before development is started, a system test plan is created. The test plan focuses on meeting the functionality specified in the requirements gathering.

The high-level design (HLD) phase focuses on system architecture and design. It provide overview of solution, platform, system, product and service/process. An integration test plan is created in this phase as well in order to test the pieces of the software systems ability to work together.

The low-level design (LLD) phase is where the actual software components are designed. It defines the actual logic for each and every component of the system. Class diagram with all the methods and relation between classes comes under LLD. Component tests are created in this phase as well.

The implementation phase is, again, where all coding takes place. Once coding is complete, the path of execution continues up the right side of the V where the test plans developed earlier are now put to use.

Coding: This is at the bottom of the V-Shape model. Module design is converted into code by developers.

Advantages of V-model

- ✓ Simple and easy to use.
- ✓ Testing activities like planning, test designing happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model.

- ✓ Proactive defect tracking – that is defects are found at early stage.
- ✓ Avoids the downward flow of the defects.
- ✓ Works well for small projects where requirements are easily understood.

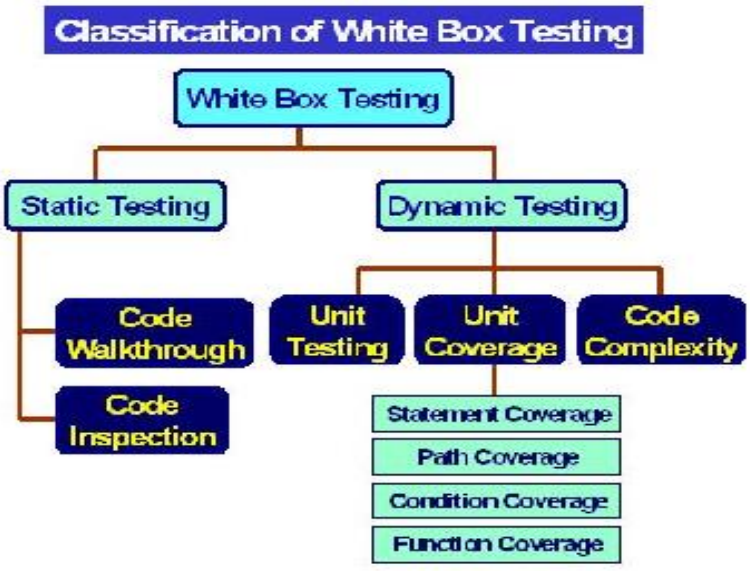
Disadvantages of V-model

- ✓ Very rigid and least flexible.
- ✓ Software is developed during the implementation phase, so no early prototypes of the software are produced.
- ✓ If any changes happen in midway, then the test documents along with requirement documents has to be updated.
- ✓ White Box Testing (WBT) is also known as Code-Based Testing or Structural Testing. White box testing is the software testing method in which internal structure is being known to tester who is going to test the software.

❖ White Box Testing

Testing based on an analysis of the internal structure of the component or system.

In this method of testing the test cases are calculated based on analysis internal structure of the system based on Code coverage, branches coverage, paths coverage, condition Coverage etc.



White box testing involves the testing by looking at the internal structure of the code & when you completely aware of the internal structure of the code then you can run your test cases & check whether the system meet requirements mentioned in the specification document.

Typically such method are used at Unit Testing of the code but this different as Unit testing done by the developer & White Box Testing done by the testers, this is learning the part of the code & finding out the weakness in the software program under test.

For tester to test the software application under test is like a white/transparent box where the inside of the box is clearly seen to the tester (as tester is aware/access of the internal structure of the code), so this method is called as White Box Testing.

The White-box testing is one of the best method to find out the errors in the software application in early stage of software development life cycle. In this process the deriving the test cases is most important part. The test case design strategy include such that all lines of the source code will be executed at least once or all available functions are executed to complete 100% code coverage of testing.

❖ Static testing

Static testing is a software testing method that involves examination of the program's code and its associated documentation but does not require the program be executed.

Static testing may be conducted manually or through the use of various software testing tools. Specific types of static software testing include code analysis, inspection, code reviews and walkthroughs.

❖ Structural testing

- ✓ The structural testing is the testing of the structure of the system or component.
- ✓ Structural testing is often referred to as ‘white box’ or ‘glass box’ or ‘clear-box testing’ because in structural testing we are interested in what is happening ‘inside the system/application’.
- ✓ In structural testing the testers are required to have the knowledge of the internal implementations of the code. Here the testers require knowledge of how the software is implemented, how it works.
- ✓ During structural testing the tester is concentrating on how the software does it. For example, a structural technique wants to know how loops in the software are working. Different test cases may be derived to exercise the loop once, twice, and many times. This may be done regardless of the functionality of the software.
- ✓ Structural testing can be used at all levels of testing. Developers use structural testing in component testing and component integration testing, especially where there is good tool support for code coverage. Structural testing is also used in system and acceptance testing, but the structures are different. For example, the coverage of menu options or major business transactions could be the structural element in system or acceptance testing.

❖ Challenges in White Box Testing

White box testing requires a sound knowledge of the program code and the programming language. This means that the developers should get intimately involved in white box testing.

Developers, in general, do not like to perform testing functions. This applies to structural testing as well as static testing methods such as reviews.

In addition, because of the timeline pressures, the programmers may not “find time” for reviews (an euphemism for wanting to do more coding).

Human tendency of a developer being unable to find the defects in his or her code As we saw earlier, most of us have blind spots in detecting errors in our own products.

Since white box testing involves programmers who write the code, it is quite possible that they may not be most effective in detecting defects in their own work products. An independent perspective could certainly help.

MULTIPLE CHOICE QUESTIONS

1. ---------- Phase is to come up with a schedule, the scope, and resource requirements for a release.
   a. Planning b. Design c. Coding d. testing
2. ---------- Phase is to figure out how to satisfy the requirements enumerated in the system requirements specification document.
   a. **Design** b. planning c. Development d. none
3. ---------- phase comprises coding the programs in the chosen programming.
   a. **Coding** b. Planning c. Design d. none
4. ---------- is meeting the requirements expected of the software, consistently and predictably.
   a. **Quality** b. Quality assurance c. quality control d. none
5. ---------- is defect prevention oriented
   a. **Quality assurance** b. Planning c. quality d. none
6. ---------- is defect detection and correction oriented
   a. **Quality control** b. quality c. design d. all

7. The purpose of---------- is to uncover defects in the system
   a. **Testing** b. verification c. validation d. all
8. ---------- is specify when that phase can be started
   a. **Entry criteria** b. Task c. Exit criteria d. none
9. ---------- specifies methods of checking that the tasks have been carried out correctly
   a. **Verification** b. validation c. a and b d. none
10. ----------model a project is divided into a set of phases
    a. **Waterfall model** b. spiral model c. v model d. none
11. White box testing is also known as----------
    a. Clear box b. glass box c. open box **d. all**
12. ----------requires only the source code of the product, not the binaries or executables
    a. **Static testing** b. white box testing c. black box testing d. none
13. ---------- takes into account the code, code structure, internal design, and how they are coded
    a. **Structural testing** b. white box testing c. static testingd. none
14. The percentage of code covered by a test is found by adopting a technique called----------
    a. **Instrumentation** b. debugging c. testing d. all
15. Total path exercised/ total number of paths in program is called----------
    a. **Path coverage** b. code coverage c. condition coverage d. all
16. Functions are easier to identify in a program and hence it is easier to write test cases to provide----------
    a. **Function coverage** b. code coverage c. path coverage d. all

17. ETVX stands for----------
    a. **Entry Task Verification eXit** b. Entity Entry Task Verification eXit c. a and b d. none
18. -------- is the output of the phase
    a. **Exit criteria** b. entry criteria c. task d. none
19. ---------- is to prevent the defects before they take shape
    a. **Proactive** b. reactive c. validation d. all
20. ----------is to find defects that affect the product and fix them as soon as they are introduced
    a. **Reactive** b. proactive c. a and b d. none

ANSWER THE FOLLOWING QUESTIONS

4 MARKS

1. Describe the Phases of Software project
2. State the Quality Assurance and Quality Control

3. DfineVerification and Validation
4. Describe waterfall Model.
5. Identify white box testing

6 MARKS

1. Explain Phases of software project?
2. Classify Testing, verification and Validation?
3. Describe the process model to represent different phases?
4. Discuss about life cycle models?

10 Marks

1. List out the life Cycle Models and describe it