ASSIGNMENT-4

1.Explain the various phases of SDLC?

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

Requirement gathering and analysis

Design

Implementation or coding

Testing

Deployment

Maintenance

1) Requirement gathering and analysis:  Business requirements are gathered in this phase. 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. The testing team follows the Software Testing Life Cycle and starts the [Test Planning](http://istqbexamcertification.com/what-is-the-purpose-and-importance-of-test-plans/) phase after the requirements analysis is completed.

2)  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.

In this phase the testers comes up with the [Test strategy](http://istqbexamcertification.com/what-are-the-test-approaches-or-strategies-in-software-testing/), where they mention what to test, how to test.

3)  Implementation / Coding:  On receiving system design documents, the work is divided in modules/units and actual coding is started. 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.

4)  [Testing](http://istqbexamcertification.com/what-is-a-software-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 all types of [functional testing](http://istqbexamcertification.com/what-is-functionality-testing-in-software/) like [unit testing](http://istqbexamcertification.com/what-is-unit-testing/), [integration testing](http://istqbexamcertification.com/what-is-integration-testing/), [system testing](http://istqbexamcertification.com/what-is-system-testing/), [acceptance testing](http://istqbexamcertification.com/what-is-acceptance-testing/) are done as well as [non-functional testing](http://istqbexamcertification.com/what-is-non-functional-testing-testing-of-software-product-characteristics/) are also done.

5)  Deployment: After successful testing the product is delivered / deployed to the customer for their use.

As soon as the product is given to the customers they will first do the [beta testing](http://istqbexamcertification.com/what-is-beta-testing/). If any changes are required or if any bugs are caught, then they will report it to the engineering team. Once those changes are made or the [bugs](http://istqbexamcertification.com/what-is-defect-or-bugs-or-faults-in-software-testing/) are fixed then the final deployment will happen.

6) 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.

2. Explain the various phases of STLC?

Ans. Phases of the software testing life cycle may very, but in general, they involve planning, preparing, conducting, and reporting. Lets take a close look at different parts of the software testing life cycle.

1. Requirements Analysis

In this first phase of the software testing life cycle, the test team reviews any requirements documents and designs to determine what is testable. By studying the requirements, the testing teams gets an understanding of the scope of testing. This phase might involve conversations with developers, designers, and stakeholders.

2. Test Planning

What to test, how the test needs to be done, and who’s going to test it… these are the things determined during the test planning phase. Once the requirements have been reviewed, it’s time to plan the testing project at a high level. A [test plan document](https://blog.testlodge.com/what-is-a-test-plan-in-software-testing/) is created during this phase. This phase gets everyone on the same page as far as how the testing project will be approached.

3. Test Case Development

The goal of this phase is to determine in detail “how” to test. [Test cases](https://blog.testlodge.com/how-to-write-test-cases-for-software-with-sample/) should be written to guide the tester through each test. If old test cases are being used, make sure they are up to date. Many tests might require test data. Prepare any test data required to run tests during this phase so that the you don’t have to spend time doing this during the tests.

4. Environment

The test environment is the configuration of software and/or hardware on which the testing team is to perform the tests. Without the test environment ready to go, you’re going to hit roadblocks. Make sure any test data necessary is entered into the system and ready to be used. It’s not uncommon for this phase to happen alongside the test case development phase.

5. Test Execution

Now that the tests are ready to go and the environment is setup, it’s time to run the tests. Using the test cases, the tester executes each test, comparing the expected results to the actual results of each test and marking it as pass/fail/skip. If the test fails, the tester should document what actually happened during the test. This phase also involves the tester logging bugs in the designated bug tracking system (determined in the test plan phase).

6. Test Reporting

Once all the tests cases are ran, the test manager should confirm all required testing have been completed. This involves an analysis of defects found and other metrics such as how many passed/failed/skipped test cases. This final phase in the software testing life Cycle might also include a retrospective on the testing project/process. This allows the team to learn and improve for future testing projects.

3. Explain Traditional s/w Development Model?

Ans. A traditional SDLC provided and continues to provide benefits for many organizations. In addition to the reason it was initiated — namely, adding structure to a previously unstructured process — the waterfall approach to system development has two primary advantages:

1. The explicit guidelines allow the use of less-experienced staff for system development, as all steps are clearly outlined. Even junior staff members who have never managed a project can follow the “recipes” in the SDLC “cookbook” to produce adequate systems. Reliance on individual expertise is reduced. Use of an SDLC can have the added benefit of providing training for junior staff, again because the sequence of steps and the tasks to be performed in each step are clearly defined.

2. The methodology promotes consistency among projects, which can reduce the cost of ongoing support and allow staff to be transferred from one project to another. Although coding techniques are not specified in a typical SDLC, the extensive documentation that is an inherent part of most methodologies simplifies ongoing maintenance by reducing reliance on the original developers for explanations of why the system was constructed as it was and which functions are included in which program modules.

4. Explain Prototype s/w Development Model?

Ans. The Software Prototyping refers to building software application prototypes which display the functionality of the product under development but may not actually hold the exact logic of the original software.

Software prototyping is becoming very popular as a software development model, as it enables to understand customer requirements at an early stage of development. It helps get valuable feedback from the customer and helps software designers and developers understand about what exactly is expected from the product under development.

Prototype is a working model of software with some limited functionality.

The prototype does not always hold the exact logic used in the actual software application and is an extra effort to be considered under effort estimation.

Prototyping is used to allow the users evaluate developer proposals and try them out before implementation.

It also helps understand the requirements which are user specific and may not have been considered by the developer during product design.

5. Explain Iterative Enhancement s/w Development Model?

Ans. An iterative [life cycle model](http://istqbexamcertification.com/what-are-the-software-development-models/) does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software for each cycle of the model.

Advantages of Iterative model:

* In iterative model we can only create a high-level design of the application before we actually begin to build the product and define the design solution for the entire product. Later on we can design and built a skeleton version of that, and then evolved the design based on what had been built.
* In iterative model we are building and improving the product step by step. Hence we can track the defects at early stages. This avoids the downward flow of the defects.
* In iterative model we can get the reliable user feedback. When presenting sketches and blueprints of the product to users for their feedback, we are effectively asking them to imagine how the product will work.
* In iterative model less time is spent on documenting and more time is given for designing.

6.Explain the various types of Maintenance phase?

Ans. There are four types of maintenance, namely, corrective, adaptive, perfective, and preventive.

Corrective maintenance-

Corrective maintenance deals with the repair of faults or defects found in day-today system functions. A defect can result due to errors in software design, logic and coding. Design errors occur when changes made to the software are incorrect, incomplete, wrongly communicated, or the change request is misunderstood. Logical errors result from invalid tests and conclusions, incorrect implementation of design specifications, faulty logic flow, or incomplete test of data. All these errors, referred to as residual errors, prevent the software from conforming to its agreed specifications. Note that the need for corrective maintenance is usually initiated by bug reports drawn by the users.

## Adaptive Maintenance

Adaptive maintenance is the implementation of changes in a part of the system, which has been affected by a change that occurred in some other part of the system. Adaptive maintenance consists of adapting software to changes in the environment such as the hardware or the operating system. The term environment in this context refers to the conditions and the influences which act (from outside) on the system. For example, business rules, work patterns, and government policies have a significant impact on the software system.

## Perfective Maintenance

Perfective maintenance mainly deals with implementing new or changed user requirements. Perfective maintenance involves making functional enhancements to the system in addition to the activities to increase the system's performance even when the changes have not been suggested by faults. This includes enhancing both the function and efficiency of the code and changing the functionalities of the system as per the users' changing needs.

Examples of perfective maintenance include modifying the payroll program to incorporate a new union settlement and adding a new report in the sales analysis system. Perfective maintenance accounts for 50%, that is, the largest of all the maintenance activities.

## Preventive Maintenance

Preventive maintenance involves performing activities to prevent the occurrence of errors. It tends to reduce the software complexity thereby improving program understandability and increasing software maintainability. It comprises documentation updating, code optimization, and code restructuring. Documentation updating involves modifying the documents affected by the changes in order to correspond to the present state of the system. Code optimization involves modifying the programs for faster execution or efficient use of storage space. Code restructuring involves transforming the program structure for reducing the complexity in source code and making it easier to understand.

Preventive maintenance is limited to the maintenance organization only and no external requests are acquired for this type of maintenance. Preventive maintenance accounts for only 5% of all the maintenance activities.

7.Diff b/w high level design and low level design?

Ans. High Level Design (HLD) is the overall system design - covering the system architecture and database design. It describes the relation between various modules and functions of the system. data flow, flow charts and data structures are covered under HLD.

Low Level Design (LLD) is like detailing the HLD. It defines the actual logic for each and every component of the system. Class diagrams with all the methods and relation between classes comes under LLD. Programs specs are covered under LLD.

8.Explain V-shaped Model?

Ans. The V - model is SDLC model where execution of processes happens in a sequential manner in V-shape. It is also known as Verification and Validation model.

V - Model is an extension of the waterfall model and is based on association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle there is a directly associated testing phase. This is a highly disciplined model and next phase starts only after completion of the previous phase.

V- Model design

Under V-Model, the corresponding testing phase of the development phase is planned in parallel. So there are Verification phases on one side of the .V. and Validation phases on the other side. Coding phase joins the two sides of the V-Model.

9.Diff b./w functional spec and business requirement specs.?

Ans. While both business and functional requirements related to the same project, there are major differences between the two. Both sets of requirements contribute to a common goal, although functional requirements are much more specific and detailed. While business requirements deal with mainly business goals and stakeholder expectations, functional requirements outline exactly how a project will support business requirements. A business requirement tells us what the future state of a project is and why the objective is worthwhile, while functional requirements tell us how we will get there. Functional requirements outline specific steps and outline how the project will be delivered. As a result, they help ensure a project is on track and are used for measuring performance.

10. Diff b/w testing and debugging?

Ans.

|  |  |
| --- | --- |
| Testing | Debugging |
| 1. Testing always starts with known conditions, uses predefined methods, and has predictable outcomes too. | 1. Debugging starts from possibly un-known initial conditions and its end cannot be predicted, apart from statistically. |
| 2. Testing can and should definitely be planned, designed, and scheduled. | 2. The procedures for, and period of, debugging cannot be so constrained. |
| 3. It proves a programmers failure. | 3. It is the programmer’s vindication. |
| 4. It is a demonstration of error or apparent correctness. | 4. It is always treated as a deductive process. |
| 5. Testing as executed should strive to be predictable, dull, constrained, rigid, and inhuman. | 5. Debugging demands intuitive leaps, conjectures, experimentation, and some freedom also. |
| 6. Much of the testing can be done without design knowledge. | 6. Debugging is impossible without detailed design knowledge. |
| 7. It can often be done by an outsider. | 7. It must be done by an insider. |
| 8. Much of test execution and design can be automated. | 8. Automated debugging is still a dream for programmers. |
| 9. Testing purpose is to find bug. | 9. Debugging purpose is to find cause of bug. |