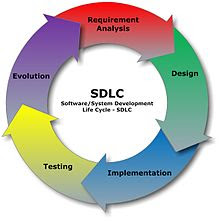
## ASSIGNMENT-4

**Q.1** Explain different phases of **SDLC**?

**Sol**:- Basically, SDLC consists of all the steps/stages of software starting from its inception to its implementation. There are in fact numerous types of [SDLC models](http://www.techknol.net/2015/11/software-development-life-cycle-sdlc-models.html) (Agile, Waterfall et al) and depending upon your requirements you can choose any of them to meet your purpose.



Given below are some phases, which are common in every SDLC model. Let us go through them to get an idea.

## ****1. Requirement Gathering and Analysis:****

Requirement gathering and analysis is the first stage of any SDLC model. This phase is basically the brainstorming phase and often consists of sub-stages like Feasibility Analysis to check how much of the idea can be put into action.

If any particular software needs to be modified, the underlying problem(s) of that software is sorted out along with finding ways to solve it. If a brand new software is going to be developed, then every minute requirement regarding that software is looked in to. This implies that this stage involves maximum research and inputs from both the company that is developing the software and the client.

## ****2. System Analysis:****

This is the second phase of SDLC where the entire system is defined in detail. In fact, it this stage wherein a detailed blueprint of various processes of the software is developed. If needed the system is divided into smaller parts to make it easier more manageable for the developers, designers, testers, project managers and other professionals who are going to work on the software in the latter stages.

## ****3. System Design:****

In this phase, the physical system is designed with the help of the logical design prepared by system analysts. The analysts and designers work together and use certain tools and software to create the overall system design, including the probable output.

## ****4. Coding:****

As the name implies, in this stage the software is coded with precision. A team of programmers are assigned by the company to work on the software. More often than not, the work is sub-divided under a sub-phase called Task Allocation, where each developer is assigned a part of the work depending on his or her skill set(s). This helps complete the coding efficiently.

## ****5. Testing:****

When the software is ready, it is sent to the testing department where Quality Analysts test it thoroughly for different errors by forming various test cases. They either test the software manually or using [automated testing tools](http://www.techknol.net/2012/05/beginners-guide-to-test-automation.html) and ensure that each and every component of the software works fine. Once the QA makes sure that the software is error-free, it goes to the next stage, which is Implementation.

## ****6. Implementation:****

This is the final stage of software development life cycle. In this stage, if the software is run on various systems by users. If it runs smoothly on these systems without any flaw, then it is considered ready to be launched.

**Q.2** Explain the various phases of STLC?

Sol:-

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

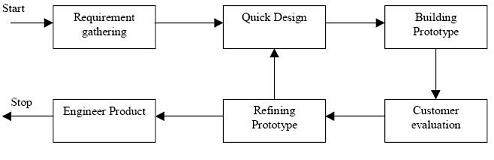
Once all the tests cases are ran, the test manager should confirm all required testing have been completed. This involves an analysis of defects found and other metrics such as how many passed/failed/skipped test cases. This final phase in the software testing life Cycle might also include a retrospective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

### Conclusion

In general, the software testing life Cycle involves planning, preparing, conducting, and reporting of tests. Some of the phases in the STLC are dependent on others, some can happen alongside each other. By understanding each phase in the STLC you can ensure your testing process is efficient and effective.

**Q.3** Explain prototype software development model?

Sol:-

  
Prototyping Model

1. The basic idea in **Prototype model** is that instead of freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements. This prototype is developed based on the currently known requirements.
2. Prototype model is a [**software development model**](http://istqbexamcertification.com/what-are-the-software-development-models/). By using this prototype, the client can get an “actual feel” of the system, since the interactions with prototype can enable the client to better understand the requirements of the desired system.
3. Prototyping is an attractive idea for complicated and large systems for which there is no manual process or existing system to help determining the requirements.

**Advantages of Prototype model:**

* Users are actively involved in the development
* Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
* Errors can be detected much earlier.
* Quicker user feedback is available leading to better solutions.
* Missing functionality can be identified easily
* Confusing or difficult functions can be identified  
  Requirements validation, Quick implementation of, incomplete, but  
  functional, application.

**Disadvantages of Prototype model:**

* Leads to implementing and then repairing way of building systems.
* Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.

**Q.4** Explain **Traditional software development model**?

Sol:-

Our development methodology comprises different phases:

-**Define the Project’s Requirements**

In the first phase, we collaborate with our clients to make sure that we understand how their business works, their goals, and their marketing plan. This is how we make sure that we completely understand the requirements of the project.

**-Create the Proposal**

Our vision for the project as a whole, including the goals for the final product, a description of the final product's functionality, all constraints (including the work environment, budget, and timeframe), and any other requirements that need to be fulfilled.

**-Develop and Approve the Functional Specifications**

The client may suggest changes, corrections, or additions to the proposal document. We always try to amend the project's scope as needed before development work actually begins, since this is the best way to avoid unnecessary expenses and stay within the project's budget.

**-Develop the Software and Conduct the Alpha Release**

Software development begins once all the project requirements and the software's architecture have been clearly defined in writing.

For each project, we identify intermediate deliverables that are shared with the client on or before defined milestone dates

Once this testing has been completed, we conduct the alpha release in which the software is given to the client for a thorough review.

**-Test for Quality Assurance and Conduct the Beta Release**

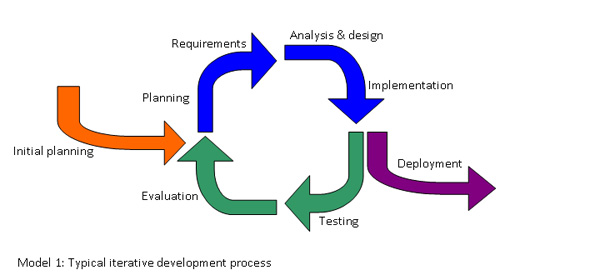
After the alpha release, we engage in extensive testing for quality assurance and conduct the beta release. All of the software's code is checked for its functionality, reliability, and consistency. Our quality assurance team looks for any deviations from our standard procedures and makes sure that the final product meets all the specifications. If any bugs or other errors are found, they are immediately corrected to comply with the requirements of the project.

**- Conduct the Final Release**

Final release, involves delivering the tailored software application to the client and deploying it on site. Our technicians install the software, making all necessary configurations for a successful product launch. We then closely monitor the system's performance so that we can offer immediate assistance if there are any problems.

**Q.5** Explain **Iterative Enhancement life cycle model**?

Sol:-



**Iterative-Model**

An iterative life cycle model 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.

Consider an iterative life cycle model which consists of repeating the following four phases in sequence:

**-A Requirements phase,** in which the requirements for the software are gathered and analysed. Iteration should eventually result in a requirements phase that produces a complete and final specification of requirements.

**-A Design phase,** in which a software solution to meet the requirements is designed. This may be a new design, or an extension of an earlier design.

**-An Implementation and Test phase,** when the software is coded, integrated and tested.

**-A Review phase,** in which the software is evaluated, the current requirements are reviewed, and changes and additions to requirements proposed.

For each cycle of the model, a decision has to be made as to whether the software produced by the cycle will be discarded, or kept as a starting point for the next cycle (sometimes referred to as [**incremental prototyping**](http://www.testingexcellence.com/incremental-model-2/)). Eventually a point will be reached where the requirements are complete and the software can be delivered, or it becomes impossible to enhance the software as required, and a fresh start has to be made.

### ****Advantages of Iterative Model:-****

* Generates working software quickly and early during the software life cycle.
* More flexible – less costly to change scope and requirements.
* Easier to test and debug during a smaller iteration.
* Easier to manage risk because risky pieces are identified and handled during its iteration.
* Each iteration is an easily managed milestone.

**Q.6** Explain the various **Types of Maintenance phase**?

Sol:-

**Software maintenance** in [software engineering](https://en.wikipedia.org/wiki/Software_engineering) is the modification of a software product after delivery to correct faults, to improve performance or other attributes.

In the late 1970s, a famous and widely cited survey study by Lientz and Swanson, exposed the very high fraction of [life-cycle costs](https://en.wikipedia.org/wiki/Whole-life_cost) that were being expended on maintenance. They categorized maintenance activities into four classes:

* **Adaptive** – modifying the system to cope with changes in the software environment ([DBMS](https://en.wikipedia.org/wiki/DBMS), [OS](https://en.wikipedia.org/wiki/Operating_system)).
* **Perfective** – implementing new or changed user requirements which concern functional enhancements to the software.
* **Corrective** – diagnosing and fixing errors, possibly ones found by users .
* **Preventive** – increasing software maintainability or reliability to prevent problems in the future .

**Q.7** What is the difference between **High Level Design and Low Level Design?**

Sol:-

**HLD**- also known as macro level/ system design...  
**LLD**- also known as micro level/ detailed design...  
  
**HLD**- 2 main components are 1>Functional architecture of application... 2>Database design...  
**LLD**- in this design,for every program logic design is done, documented and unit test plan created...  
  
**HLD**- participants are design team, review team and client...  
**LLD**- participants are only design team...  
  
**HLD**- input criteria is Software Requirement Specification(SRS)... Output criteria is database design, functional design and review record...  
**LLD**- input criteria is HLD, which are reviewed and authorized... Output criteria is Program specification and unit test plan...

**Q.8** Explain **V-shaped model**?

Sol:-

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.

The below figure illustrates the different phases in V-Model of SDLC.



## Verification Phases

Following are the Verification phases in V-Model:

* **Business Requirement Analysis:** This is the first phase in the development cycle where the product requirements are understood from the customer perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirement. This is a very important activity and need to be managed well, as most of the customers are not sure about what exactly they need. The acceptance test design planning is done at this stage as business requirements can be used as an input for acceptance testing.
* **System Design:** Once you have the clear and detailed product requirements, it.s time to design the complete system. System design would comprise of understanding and detailing the complete hardware and communication setup for the product under development. System test plan is developed based on the system design. Doing this at an earlier stage leaves more time for actual test execution later.
* **Architectural Design:** Architectural specifications are understood and designed in this phase. Usually more than one technical approach is proposed and based on the technical and financial feasibility the final decision is taken. System design is broken down further into modules taking up different functionality. This is also referred to as High Level Design (HLD).

The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood and defined in this stage. With this information, integration tests can be designed and documented during this stage.

* **Module Design:**In this phase the detailed internal design for all the system modules is specified, referred to as Low Level Design (LLD). It is important that the design is compatible with the other modules in the system architecture and the other external systems. Unit tests are an essential part of any development process and helps eliminate the maximum faults and errors at a very early stage. Unit tests can be designed at this stage based on the internal module designs.

## Coding Phase

The actual coding of the system modules designed in the design phase is taken up in the Coding phase. The best suitable programming language is decided based on the system and architectural requirements. The coding is performed based on the coding guidelines and standards. The code goes through numerous code reviews and is optimized for best performance before the final build is checked into the repository.

## Validation Phases

Following are the Validation phases in V-Model:

* **Unit Testing:** Unit tests designed in the module design phase are executed on the code during this validation phase. Unit testing is the testing at code level and helps eliminate bugs at an early stage, though all defects cannot be uncovered by unit testing.
* **Integration Testing:** Integration testing is associated with the architectural design phase. Integration tests are performed to test the coexistence and communication of the internal modules within the system.
* **System Testing:** System testing is directly associated with the System design phase. System tests check the entire system functionality and the communication of the system under development with external systems. Most of the software and hardware compatibility issues can be uncovered during system test execution.
* **Acceptance Testing:** Acceptance testing is associated with the business requirement analysis phase and involves testing the product in user environment. Acceptance tests uncover the compatibility issues with the other systems available in the user environment. It also discovers the non functional issues such as load and performance defects in the actual user environment.

**Q.9** What is the difference between **Testing** and **Debugging**?

Sol:-

|  |
| --- |
| [testing and debugging, difference in testing and debugging in software testing interview wikipedia](http://3.bp.blogspot.com/-pZMJUFF_Xmo/T-YE67H3T4I/AAAAAAAAAV0/8eviG5jZPG8/s1600/Testing_vs_Debugging.JPG) |
| Testing V/s Debugging |

**Q.10** What is the difference between **functional specification** and **business requirement specification**?

Sol:-

## Difference-Between-Business-And-Functional-Requirements