**Assignment 4**

Q Explain the various phases of SDLC?

ANSWER System Development Life Cycle (SDLC) is a series of six main phases to create a hardware system only, a software system only or a combination of both to meet or exceed customer’s expectations.

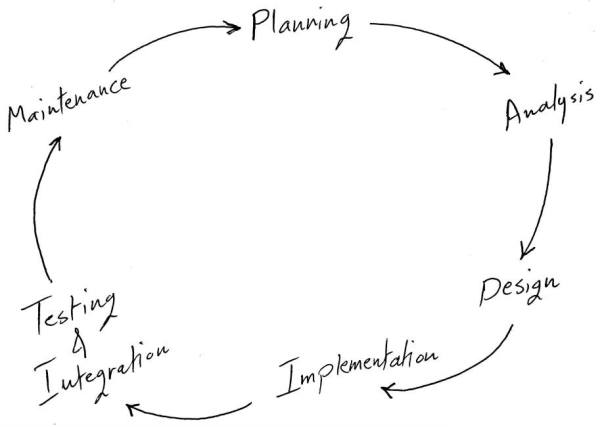
**System** is a broad and a general term, and as per to Wikipedia; “A **system** is a set of interacting or interdependent components forming an integrated whole” it’s a term that can be used in different industries, therefore [Software Development Life Cycle](https://airbrake.io/blog/insight/what-is-the-software-development-life-cycle) is a limited term that explains the phases of creating a software component that integrates with other software components to create the whole system.

Some more specific takes on SDLC include:

* [Rapid Application Development](https://airbrake.io/blog/sdlc/rapid-application-development)
* [Spiral Model](https://airbrake.io/blog/sdlc/spiral-model)
* [Waterfall Model](https://airbrake.io/blog/sdlc/waterfall-model)
* [Big Bang Model](https://airbrake.io/blog/sdlc/big-bang-model)
* [Iterative Model](https://airbrake.io/blog/sdlc/iterative-model)

Below we’ll take a general look on System Development Life Cycle phases, bearing in mind that each system is different from the other in terms of complexity, required components and expected solutions and functionalities:

System Development Life Cycle Phases:

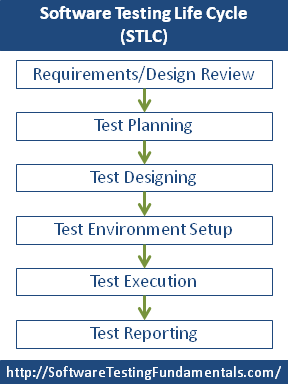


Q WHAT ARE THE VARIOUS PHASES  OF STLC?

ANSWER **Software Testing Life Cycle (STLC)**defines the steps/ stages/ phases in testing of software. However, there is no fixed standard STLC in the world and it basically varies as per the following:

* [Software Development Life Cycle](http://softwaretestingfundamentals.com/software-development-life-cycle/)
* Whims of the Management

Nevertheless, Software Testing Life Cycle, in general, comprises of the following phases:



Q EXPLAIN TRADITIONAL SOFTWARE DEVELOPMENT MODEL?

ANSWER The selection of model has very high impact on the testing that is carried out. It will define the what, where and when of our planned testing, influence regression testing and largely determines which test techniques to use.

There are various Software development models or methodologies. They are as follows:

1. [**Waterfall model**](http://istqbexamcertification.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/)
2. [**V model**](http://istqbexamcertification.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/)
3. [**Incremental model**](http://istqbexamcertification.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/)
4. [**RAD model**](http://istqbexamcertification.com/what-is-rad-model-advantages-disadvantages-and-when-to-use-it/)
5. [**Agile model**](http://istqbexamcertification.com/what-is-agile-model-advantages-disadvantages-and-when-to-use-it/)
6. [**Iterative model**](http://istqbexamcertification.com/what-is-iterative-model-advantages-disadvantages-and-when-to-use-it/)
7. [**Spiral model**](http://istqbexamcertification.com/what-is-spiral-model-advantages-disadvantages-and-when-to-use-it/)
8. [**Prototype model**](http://istqbexamcertification.com/what-is-prototype-model-advantages-disadvantages-and-when-to-use-it/)

Q EXPLAIN PROTOTYPE SOFTWARE DEVELOPMENT MODEL?

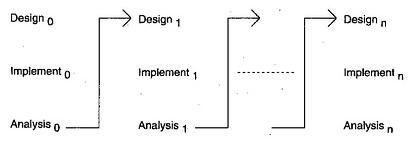
ANSWER The basic idea in **Prototype model** is that instead of freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements. This prototype is developed based on the currently known requirements. Prototype model is a [**software development 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.  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.

Q EXPLAIN ITERATIVE ENHANCEMENT LIFE CYCLE MODEL?

ANSWER 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.

in the diagram above when we work **iteratively**we create rough product or product piece in one iteration, then review it and improve it in next iteration and so on until it’s finished. As shown in the image above, in the first iteration the whole painting is sketched roughly, then in the second iteration colors are filled and in the third iteration finishing is done. Hence, in iterative model the whole product is developed step by step.

**Diagram of Iterative model:**



Q EXPLAIN THE VARIOUS TYPES OF MAINTAINCE PHASE ?

ANSWER There are four types of maintenance, namely, corrective, adaptive, perfective, and preventive. Corrective maintenance is concerned with fixing errors that are observed when the software is in use. Adaptive maintenance is concerned with the change in the software that takes place to make the software adaptable to new environment such as to run the software on a new [operating system](http://ecomputernotes.com/fundamental/disk-operating-system/what-is-operating-system). Perfective maintenance is concerned with the change in the software that occurs while adding new functionalities in the software. Preventive maintenance involves implementing changes to prevent the occurrence of errors. The distribution of types of maintenance by type and by percentage of time consumed.

Q WHAT IS THE DIFFERENCE BETWEEN HIGH LEVEL DESIGN AND LOW LEVEL DESIGN?

ANSWER

# High Level Design or System Design (HLD)

High  level Design gives the overall System Design in terms of **Functional Architecture and Database design**. This is very useful for the developers to understand the flow of the system.  In this phase design team, review team (testers) and customers plays a major role. For this the entry criteria are the requirement document that is SRS. And the exit criteria will be HLD, projects standards, the functional design documents, and the database design document.

**Low  Level Design (LLD)**

During the detailed phase, the view of the application developed during the high level design is broken down into modules and programs. Logic design is done for every program and then documented as **program specifications**. For every program, a **unit test** plan is created.

The entry criteria for this will be the HLD document. And the exit criteria will the program specification and unit test plan (LLD).

Q EXPLAIN V – SHAPED MODEL ?

## ANSWER

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.

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



Q DIFFERENCE BETWEEN TESTING AND DEBUGGING ?

ANSWER

|  |  |
| --- | --- |
| 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. |