Software Development Life Cycle

The development models are the various processes or methodologies that are being selected for the development of the project depending on the project’s aims and goals. There are many development life cycle models that have been developed in order to achieve different required objectives. The models specify the various stages of the process and the order in which they are carried out.

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. Here are a few.

# 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.

**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 on-going 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.

# 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.



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

**Requirements:**  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.

* In system testing the behaviour of whole system/product is tested as defined by the scope of the development project or product.
* It may include tests based on risks and/or requirement specifications, business process, use cases, or other high level descriptions of system behaviour, interactions with the operating systems, and system resources.
* System testing is most often the final test to verify that the system to be delivered meets the specification and its purpose.
* System testing is carried out by specialist testers or independent testers.
* System testing should investigate both functional and non-functional requirements of the testing.

**The high-level design (HLD)** phase focuses on system architecture and design. It provides an overview of solution, platform, system, product and service/process. An integration test plan is created (see notes below) 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.

**When to use the V-model:**

* The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed.
* The V-Shaped model should be chosen when ample technical resources are available with needed technical expertise.

High confidence of customer is required for choosing the V-Shaped model approach. Since, no prototypes are produced, there is a very high risk involved in meeting customer expectations.

### Integration Testing

* Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems.
* Integration testing is done by a specific integration tester or test team.
	+ **Big Bang integration testing:**
		- In Big Bang integration testing all components or modules are integrated simultaneously, after which everything is tested as a whole.
		- Big Bang testing has the advantage that everything is finished before integration testing starts.
		- The major disadvantage is that in general it is time consuming and difficult to trace the cause of failures because of this late integration.
	+ **Incremental testing:**
		- Another extreme is that all programmers are integrated one by one, and a test is carried out after each step.
		- The incremental approach has the advantage that the defects are found early in a smaller assembly when it is relatively easy to detect the cause.
		- A disadvantage is that it can be time-consuming since stubs and drivers have to be developed and used in the test.
		- Within incremental integration testing  a range of possibilities exist, partly depending on the system architecture:

**– Top down:** Testing takes place from top to bottom, following the control flow or architectural structure (e.g. starting from the GUI or main menu). Components or systems are substituted by stubs.

**– Bottom up:**Testing takes place from the bottom of the control flow upwards. Components or systems are substituted by drivers.

**– Functional incremental:**Integration and testing takes place on the basis of the functions and functionalities, as documented in the functional specification.

# Incremental model

In incremental model the whole requirement is divided into various builds. Multiple development cycles take place here, making the life cycle a “multi-waterfall” cycle..  Cycles are divided up into smaller, more easily managed modules.  Each module passes through the requirements, design, implementation and testing phases. A working version of software is produced during the first module, so you have working software early on during the software life cycle. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is achieved.

Eg.

 

In the diagram above when we work **incrementally**we are adding piece by piece but expect that each piece is fully finished. Thus keep on adding the pieces until it’s complete.



**Advantages of Incremental 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.
* Customer can respond to each built.
* Lowers initial delivery cost.
* Easier to manage risk because risky pieces are identified and handled during it’d iteration.

**Disadvantages of Incremental model:**

* Needs good planning and design.
* Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
* Total cost is higher than waterfall.

**When to use the Incremental model:**

* Requirements of the complete system are clearly defined and understood.
* Major requirements must be defined; however, some details can evolve with time.
* There is a need to get a product to the market early.
* A new technology is being used
* Resources with needed skill set are not available
* There are some high risk features and goals.

# Spiral model

The spiral model is similar to the incremental model, with more emphasis placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). In the baseline spiral, starting in the planning phase, requirements are gathered and risk is assessed. Each subsequent spiral builds on the baseline spiral. **Requirements** are gathered during the planning phase.  In the**risk analysis phase**, a process is undertaken to identify risk and alternate solutions.  A prototype is produced at the end of the risk analysis phase.

Software is produced in the **engineering phase**, along with testing at the end of the phase.  The **evaluation phase** allows the customer to evaluate the output of the project to date before the project continues to the next spiral.



**Advantages of Spiral model:**

* High amount of risk analysis hence, avoidance of Risk is enhanced.
* Good for large and mission-critical projects.
* Strong approval and documentation control.
* Additional Functionality can be added at a later date.
* Software is produced early in the software life cycle.

**Disadvantages of Spiral model:**

* Can be a costly model to use.
* Risk analysis requires highly specific expertise.
* Project’s success is highly dependent on the risk analysis phase.
* Doesn’t work well for smaller projects.

**When to use Spiral model:**

* When costs and risk evaluation is important
* For medium to high-risk projects
* Long-term project commitment unwise because of potential changes to economic priorities
* Users are unsure of their needs
* Requirements are complex
* New product line
* Significant changes are expected (research and exploration)