ACTIVITY-6

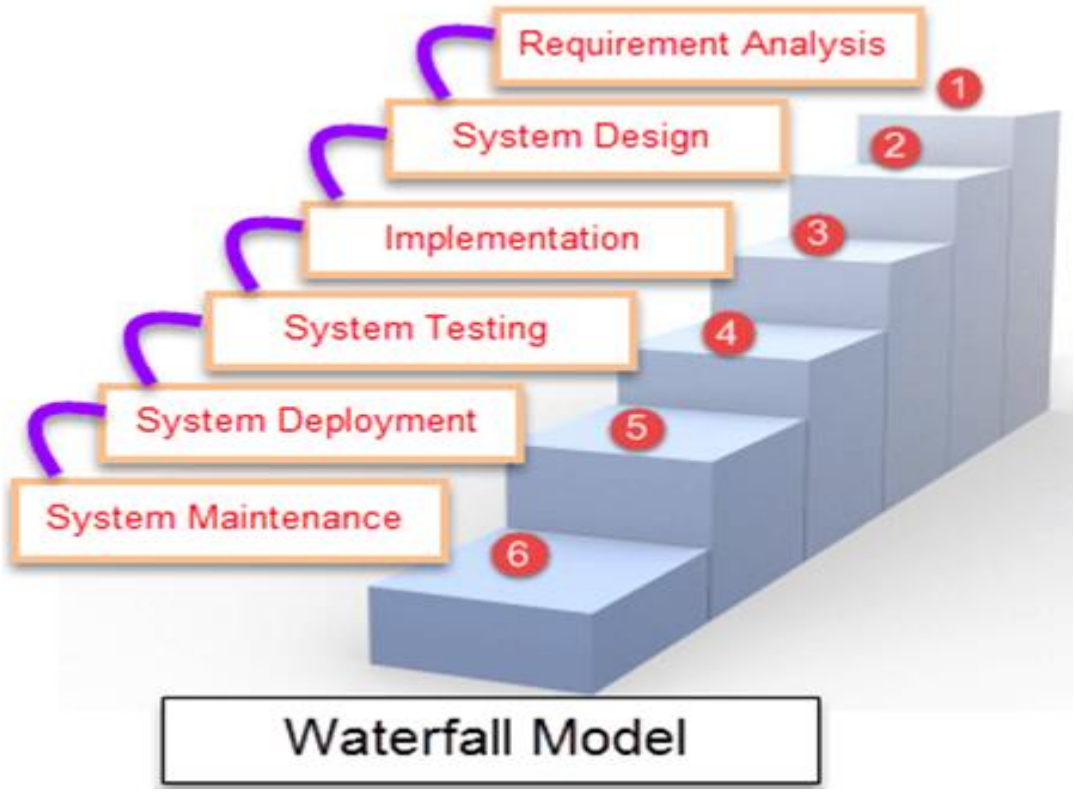
Abimanyu M

EBEON0323767152

BATCH NO:2022-9615

SDLC – WATERFALL MODEL

The software development lifecycle (SDLC) is **the cost-effective and time-efficient process that development teams use to design and build high-quality software**. The goal of SDLC is to minimize project risks through forward planning so that software meets customer expectations during production and beyond.



Some Circumstances where the use of the Waterfall model is most suited are:

- When the requirements are constant and not changed regularly.

- A project is short
- The situation is calm
- Where the tools and technology used is consistent and is not changing
- When resources are well prepared and are available to use.

Different phases	Activities performed in each stage
Requirement Gathering stage	● During this phase, detailed requirements of the software system to be developed are gathered from client
Design Stage	● Plan the programming language, for Example Java, PHP, .net ● or database like Oracle, MySQL, etc. ● Or other high-level technical details of the project
Built Stage	● After design stage, it is built stage, that is nothing but coding the software
Test Stage	● In this phase, you test the software to verify that it is built as per the specifications given by the client.
Deployment stage	● Deploy the application in the respective environment
Maintenance stage	● Once your system is ready to use, you may later require change the code as per customer request

ADVANTAGES:

- This paradigm is straightforward and easy to understand. Because of this, teamwork is incredibly simple and everyone is on the same page.
- With this model, deadlines are met with ease. This is because the team has previously been given precise instructions and no time is lost trying to figure out how to proceed.
- Since this type of architecture involves a lot of paperwork, it is better suited for larger projects with greater teams.

DISADVANTAGES

- Making adjustments to the product at a later stage in the project is practically difficult, which is one of this model's main drawbacks. This is because the steps are tied to one another in a sequential process, thus changing anything would be exceedingly difficult.
- The inability of the stakeholders and customers to utilise or view the product right away is another drawback of this approach. The customer will have to wait a few months before they can view the goods, which occasionally causes them to feel uncomfortable or disappointed.
- As the product cannot be altered at a later stage of the development process, there is also no space for error. Therefore, extensive investigation is required.
- A paradigm like this is not appropriate for small teams because it necessitates excessive research, which raises the cost.

SDLC-ITERATION MODEL

The iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).

Iterative and Incremental development is a combination of both iterative design or iterative method and incremental build model for development. "During software development, more than

one iteration of the software development cycle may be in progress at the same time." This process may be described as an "evolutionary acquisition" or "incremental build" approach."

The following illustration is a representation of the Iterative and Incremental model −

Iterative and Incremental development is a combination of both iterative design or iterative method and incremental build model for development. "During software development, more than one iteration of the software development cycle may be in progress at the same time." This process may be described as an "evolutionary acquisition" or "incremental build" approach."

The advantages of the Iterative and Incremental SDLC Model are as follows −

- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.
- Testing and debugging during smaller iteration is easy.

- Risks are identified and resolved during iteration; and each iteration is an easily managed milestone.
- Easier to manage risk - High risk part is done first.
- With every increment, operational product is delivered.
- Issues, challenges and risks identified from each increment can be utilized/applied to the next increment.
- Risk analysis is better.
- It supports changing requirements.
- Initial Operating time is less.
- Better suited for large and mission-critical projects.
- During the life cycle, software is produced early which facilitates customer evaluation and feedback.
- DISADVANTAGE
- More resources may be required.
- Although cost of change is lesser, but it is not very suitable for changing requirements.
- More management attention is required.
- System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.
- Defining increments may require definition of the complete system.
- Not suitable for smaller projects.
- Management complexity is more.
- End of project may not be known which is a risk.
- Highly skilled resources are required for risk analysis.
- Projects progress is highly dependent upon the risk analysis phase.

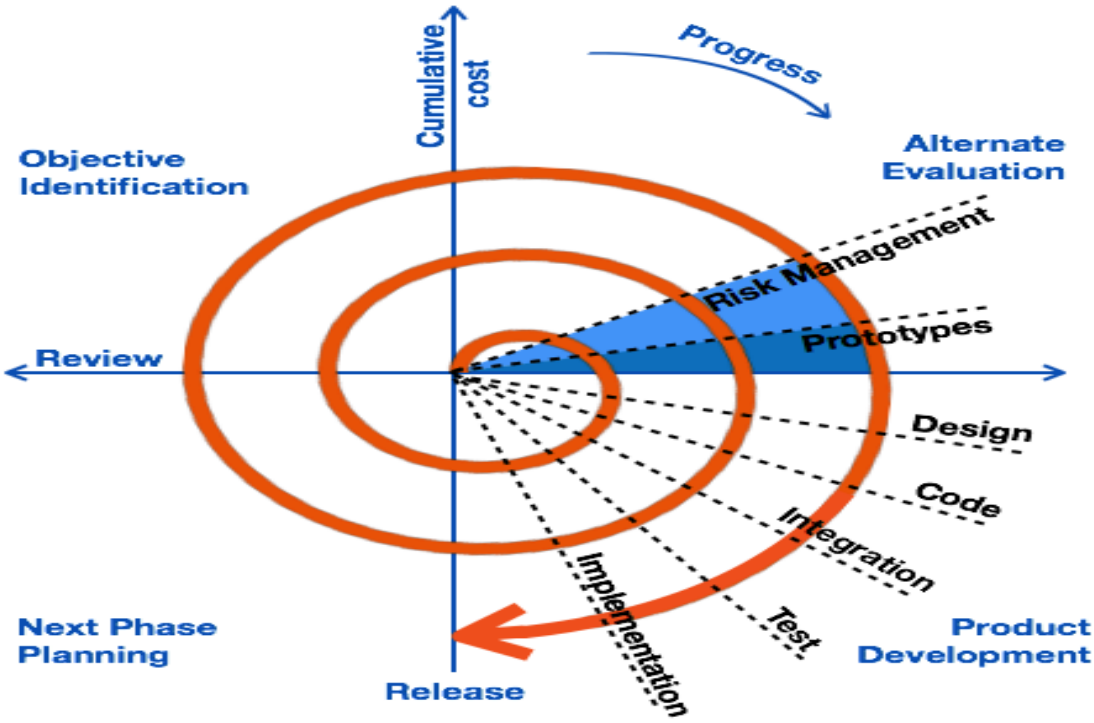
SDLC-SPIRAL MODEL

SPIRAL DESIGN:

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model. This Spiral model is a combination of iterative development process model and sequential linear development model i.e., the waterfall model with a very high emphasis on risk analysis. It allows incremental releases of the product or incremental refinement through each iteration around the spiral.

DESIGN:

The Design phase starts with the conceptual design in the baseline spiral and involves architectural design, logical design of modules, physical product design and the final design in the subsequent spirals.



SPIRAL MODEL APPLICATION:

The Spiral Model is widely used in the software industry as it is in sync with the natural development process of any product, i.e., learning with maturity which involves minimum risk for the customer as well as the development firms.

The following pointers explain the typical uses of a Spiral Model −

- When there is a budget constraint and risk evaluation is important.
- For medium to high-risk projects.

- Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
- The customer is not sure of their requirements which is usually the case.
- Requirements are complex and need evaluation to get clarity.
- A new product line should be released in phases to get enough customer feedback.
- Significant changes are expected in the product during the development cycle.

SPIRAL MODEL - PROS AND CONS:

The advantage of spiral lifecycle model is that it allows elements of the product to be added in, when they become available or known. This ensures that there is no conflict with previous requirements and design.

This method is consistent with approaches that have multiple software builds and releases which allows making an orderly transition to a maintenance activity. Another positive aspect of this method is that the spiral model forces an early user involvement in the system development effort.

On the other hand, it takes very strict management to complete such products and there is a risk of running the spiral in an indefinite loop. So, the discipline of change and the extent of taking change requests is very important to develop and deploy the product successfully.

ADVANTAGE:

- Changing requirements can be accommodated.
- Allows extensive use of prototypes.
- Requirements can be captured more accurately.
- Users see the system early.
- Development can be divided into smaller parts and the risky parts can be developed earlier which helps in better risk management.

DISADVANTAGE:

- Management is more complex.
- The end of the project may not be known early.
- Not suitable for small or low risk projects and could be expensive for small projects.
- Process is complex
- The spiral may go on indefinitely.

- A large number of intermediate stages require excessive documentation.

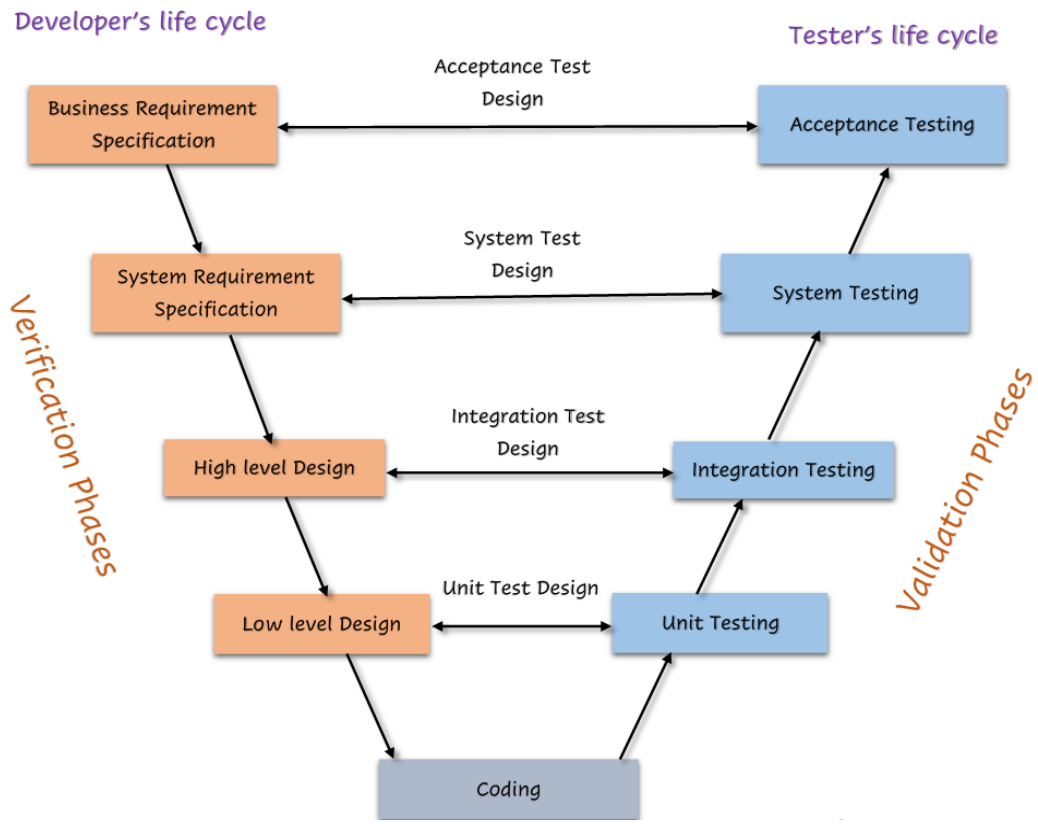
SDLC V-MODEL

What is V Model?

A contemporary traditional **software development model** is **'V-Model'**. 'V' stands for **verification and validation** and is an extension of the **Waterfall model**

The crux of the V model establishes an association between each phase of testing with that of development. The phases of testing are categorized as **"Validation Phase"** and that of development as **"Verification Phase"**. Therefore, for each phase of development there's a corresponding test activity planned.

Architecture of V Model



Phases Of V Model:

Verification Phases:

- **Requirement Analysis:** The preliminary step in software development is to gather requirements. Requirements comprise of business requirements that are to be met during the process of software development.
- Business requirement analysis is to understand an aspect from a customer's perspective by stepping into their shoes to completely analyze the functionality of an application from a user's point of view. Henceforth an **acceptance criteria** layout is prepared to correlate the tasks done during the **development process** with the outcome of the overall effort.
- **System Design:** It comprises of creating a layout of the system/application design that is to be developed. System design is aimed at writing detailed hardware and software specifications.
- System design is further segregated into subcategories as follows:

1. **Architectural Design: Architectural** design is concerned with drafting the technical methodologies to be adopted regarding completion of software development objectives. Architectural design is often termed as 'high-level design' which is aimed at providing an overview of solution, platform, system, product and service.
2. **Module Design:** Module design is known as 'low-level design' which is aimed at defining the logic upon which the system shall be built. In this stage we try to depict the relation between the modules and the order in which they interact.

Validation Phases:

- **Unit Testing Phase:** Unit tests are supposed to verify singles modules and remove bugs, if they exist. A **unit test** is simply executing a piece of code to verify whether it delivers the desired functionality.
- **Integration Testing:** The term **'integration testing'** refers to collaborating pieces of code together to verify that they perform as a single entity.
- **System Testing: System Testing** is performed when the complete system is ready, the application is then run on the target environment in which it must operate, and a conclusion is drawn to figure out whether the system can perform efficiently with least response time.
- **User Acceptance Testing:** The **user acceptance test plan** is prepared during the requirement analysis phase because when the software is ready to be delivered, it is tested against a set of tests that must be met in order to certify that the product has achieved the target it was intended to.

Advantages of V-model:

- Simple and easy to use.
- Testing activities like planning, test designing happens well before coding. ...
- 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.

Demerits of V-Model:

Lacks flexibility when it comes to making slight modifications in any of the phases, as the activities are planned and occur in a sequence

- Code is written in the implementation phase after preliminary steps for finalizing that phase is over. Hence there isn’t any prototype ready prior to the implementation stage.
- On the event of any change, it must be update in both test and requirements document