SDLC

Software Development Life Cycle

Agenda:

- Introduction to Processes.
- SDLC
- SDLC Models
- Testing
- STLC

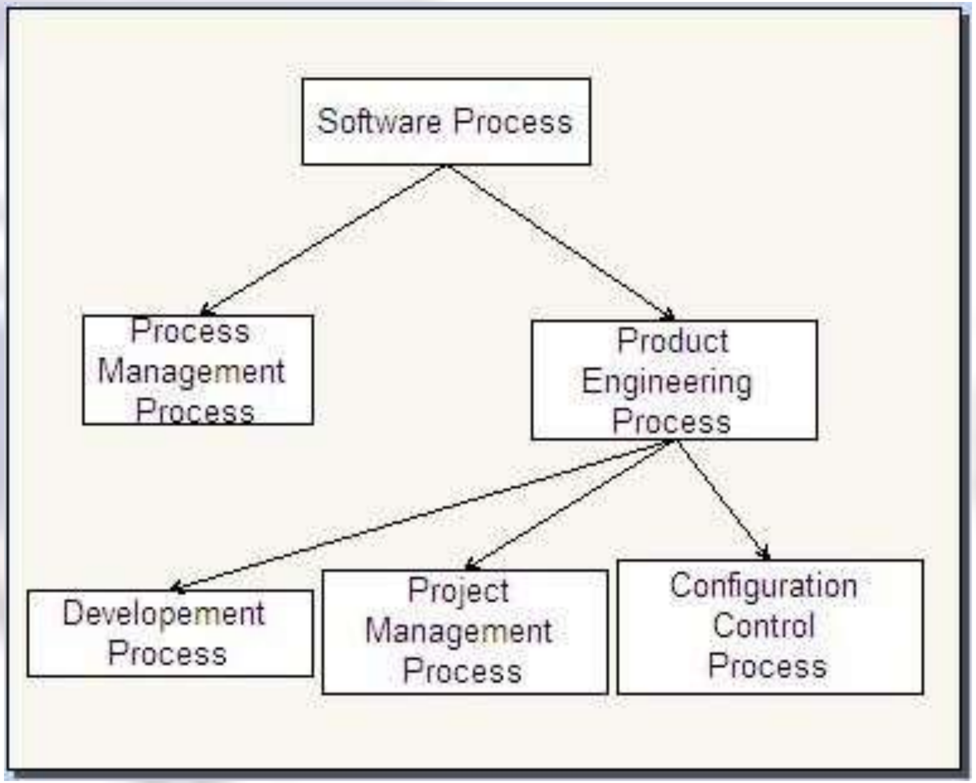
Process :

A process is the sequence of steps executed to achieve a goal. A process is defined by cycles. Similar to a project, a process also has a beginning, middle, and end; however, this cycle repeats itself over an average period of time.

Project :

A project is defined by a fixed time, scope, and resources. When implementing a project, the goal is to execute change, usually drastic, and to incorporate that change into the day-to-day processes of the company.

Software Process:

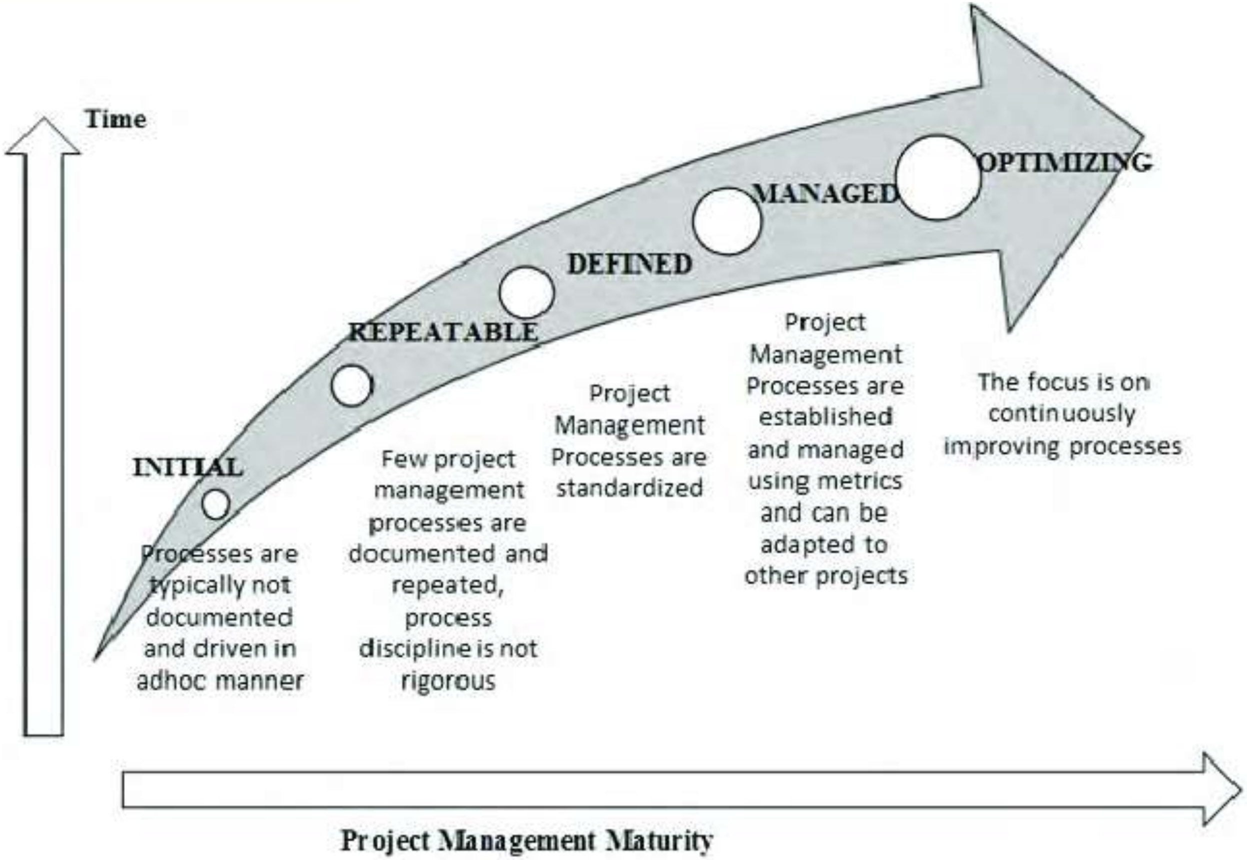

Software Process
Process Management Process
Product Engineering Process
Developement Process
Project Management Process
Configuration Control Process

The processes that deal with the technical and management issues of software development are collectively called the **software process.**

Capability Maturity Model:

- A bench-mark for measuring the maturity of an organization’s software process.
- CMM defines 5 levels of process maturity based on certain Key Process Areas (KPA).

Capability Maturity Model

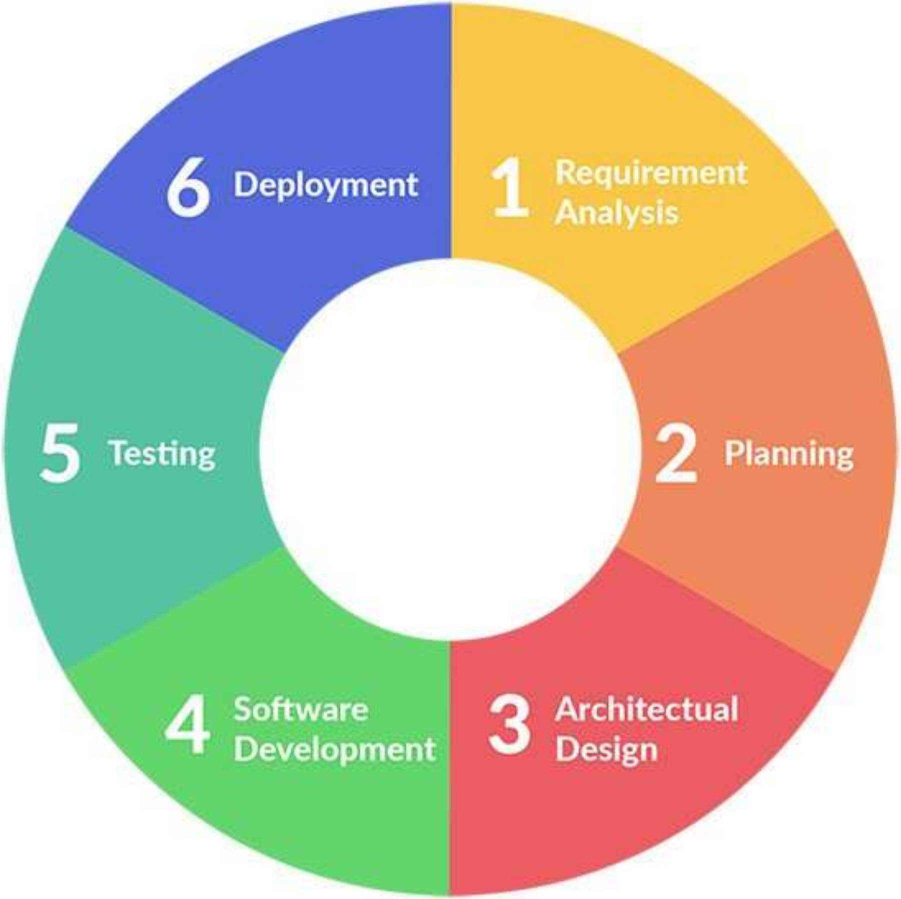


Time
INITIAL
REPEATABLE
DEFINED
MANAGED
OPTIMIZING
Processes are typically not documented and driven in adhoc manner
Few project management processes are documented and repeated, process discipline is not rigorous
Project Management Processes are standardized
Project Management Processes are established and managed using metrics and can be adapted to other projects
The focus is on continuously improving processes
Project Management Maturity

SDLC: Software Development Life Cycle

Software Development Life Cycle (SDLC) is a framework that defines the steps involved in the development of software at each phase. It covers the detailed plan for building, deploying and maintaining the software.

SDLC defines the complete cycle of development i.e. all the tasks involved in planning, creating, testing, and deploying a Software Product


1 Requirement Analysis
2 Planning
3 Architectual Design
4 Software Development
5 Testing
6 Deployment

Software Development Life Cycle Process

SDLC is a process that defines the various stages involved in the development of software for delivering a high-quality product. SDLC stages cover the complete life cycle of a software i.e. from inception to retirement of the product.

Adhering to the SDLC process leads to the development of the software in a systematic and disciplined manner.

Purpose:

Purpose of SDLC is to deliver a high-quality product which is as per the customer’s requirement.

SDLC has defined its phases as, Requirement gathering, Designing, Coding, Testing, and Maintenance. It is important to adhere to the phases to provide the Product in a systematic manner.

For Example, A software has to be developed and a team is divided to work on a feature of the product and is allowed to work as they want. One of the developers decides to design first whereas the other decides to code first and the other on the documentation part.
This will lead to project failure because of which it is necessary to have a good knowledge and understanding among the team members to deliver an expected product.

SDLC Phases:

- Requirement gathering and analysis
- Design
- Implementation or coding
- Testing
- Deployment
- Maintenance

1) Requirement Gathering and Analysis

During this phase, all the relevant information is collected from the customer to develop a product as per their expectation.

Any ambiguities must be resolved in this phase only.

Business analyst and Project Manager set up a meeting with the customer to gather all the information like what the customer wants to build, who will be the end-user, what is the purpose of the product.

Before building a product a core understanding or knowledge of the product is very important.

2) Design

In this phase, the requirement gathered in the SRS document is used as an input and software architecture that is used for implementing system development is derived.

3) Implementation or Coding

Implementation/Coding starts once the developer gets the Design document. The Software design is translated into source code. All the components of the software are implemented in this phase.

4) Testing

Testing starts once the coding is complete and the modules are released for testing.

In this phase, the developed software is tested thoroughly and any defects found are assigned to developers to get them fixed.

Retesting, regression testing is done until the point at which the software is as per the customer’s expectation.

Testers refer SRS document to make sure that the software is as per the customer’s standard.

5) Deployment

Once the product is tested, it is deployed in the production environment or first UAT (User Acceptance testing) is done depending on the customer expectation.

In the case of UAT, a replica of the production environment is created and the customer along with the developers does the testing. If the customer finds the application as expected, then sign off is provided by the customer to go live.

6) Maintenance

After the deployment of a product on the production environment, maintenance of the product i.e. if any issue comes up and needs to be fixed or any enhancement is to be done is taken care by the developers.

Software Development Life Cycle Models

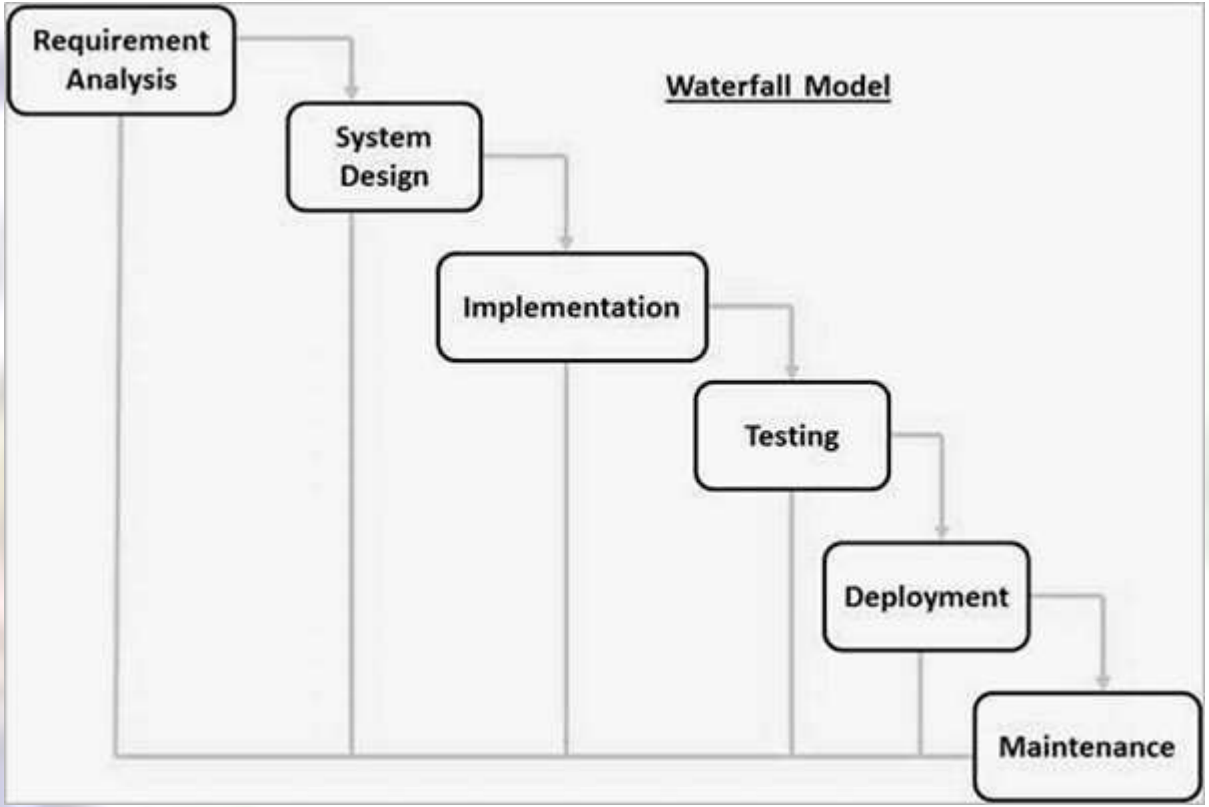
A software life cycle model is a descriptive representation of the software development cycle.

SDLC models might have a different approach but the basic phases and activity remain the same for all the models.

1) Waterfall Model

Waterfall model is the very first model that is used in SDLC. It is also known as the linear sequential model.

In this model, the outcome of one phase is the input for the next phase. Development of the next phase starts only when the previous phase is complete.



Waterfall Model
Requirement Analysis
System Design
Implementation
Testing
Deployment
Maintenance

Advantages of the Waterfall Model:

- Waterfall model is the simple model which can be easily understood and is the one in which all the phases are done step by step.

- Deliverables of each phase are well defined, and this leads to no complexity and makes the project easily manageable.

Disadvantages of Waterfall model:

•Waterfall model is time-consuming & cannot be used in the short duration projects as in this model a new phase cannot be started until the ongoing phase is completed.

•Waterfall model cannot be used for the projects which have uncertain requirement or wherein the requirement keeps on changing.

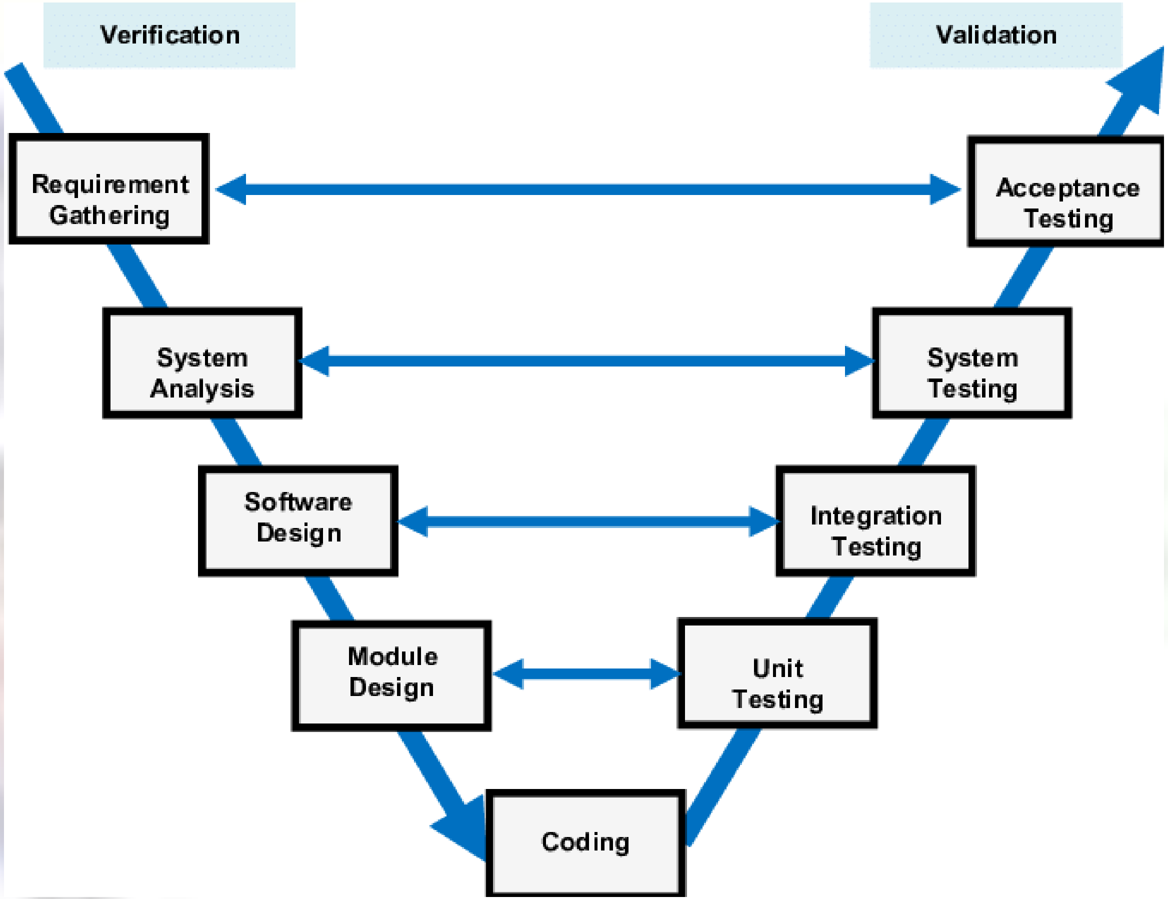
•This model expects the requirement to be clear in the requirement gathering and analysis phase itself and any change in the later stages would lead to cost higher as the changes would be required in all the phases.

2) V-Shaped Model

V- Model is also known as Verification and Validation Model.

In this model Verification & Validation goes hand in hand i.e. development and testing goes parallel.

V model and waterfall model are the same except that the test planning and testing start at an early stage in V-Model.


Verification
Validation
Requirement Gathering
Acceptance Testing
System Analysis
System Testing
Software Design
Integration Testing
Module Design
Unit Testing
Coding

a) Verification Phase:

(i) Requirement Analysis:

In this phase, all the required information is gathered & analyzed. Verification activities include reviewing the requirements.

(ii) System Design:

Once the requirement is clear, a system is designed i.e. architecture, components of the product are created and documented in a design document.

(iii) High-Level Design:

High-level design defines the architecture/design of modules. It defines the functionality between the two modules.

(iv) Low-Level Design:

Low-level Design defines the architecture/design of individual components.

(v) Coding:

Code development is done in this phase.

b) Validation Phase:

(i) Unit Testing:

Unit testing is performed using the unit test cases that are designed and is done in the Low-level design phase. Unit testing is performed by the developer itself. It is performed on individual components which lead to early defect detection.

(ii) Integration Testing:

Integration testing is performed using integration test cases in High-level Design phase. Integration testing is the testing that is done on integrated modules. It is performed by testers.

(iii) System Testing:

System testing is performed in the System Design phase. In this phase, the complete system is tested i.e. the entire system functionality is tested.

(iv) Acceptance Testing:

Acceptance testing is associated with the Requirement Analysis phase and is done in the customer’s environment.

Advantages of V – Model:

- It is a simple and easily understandable model.
- V –model approach is good for smaller projects wherein the requirement is defined and it freezes in the early stage.
- It is a systematic and disciplined model which results in a high-quality product.

Disadvantages of V-Model:

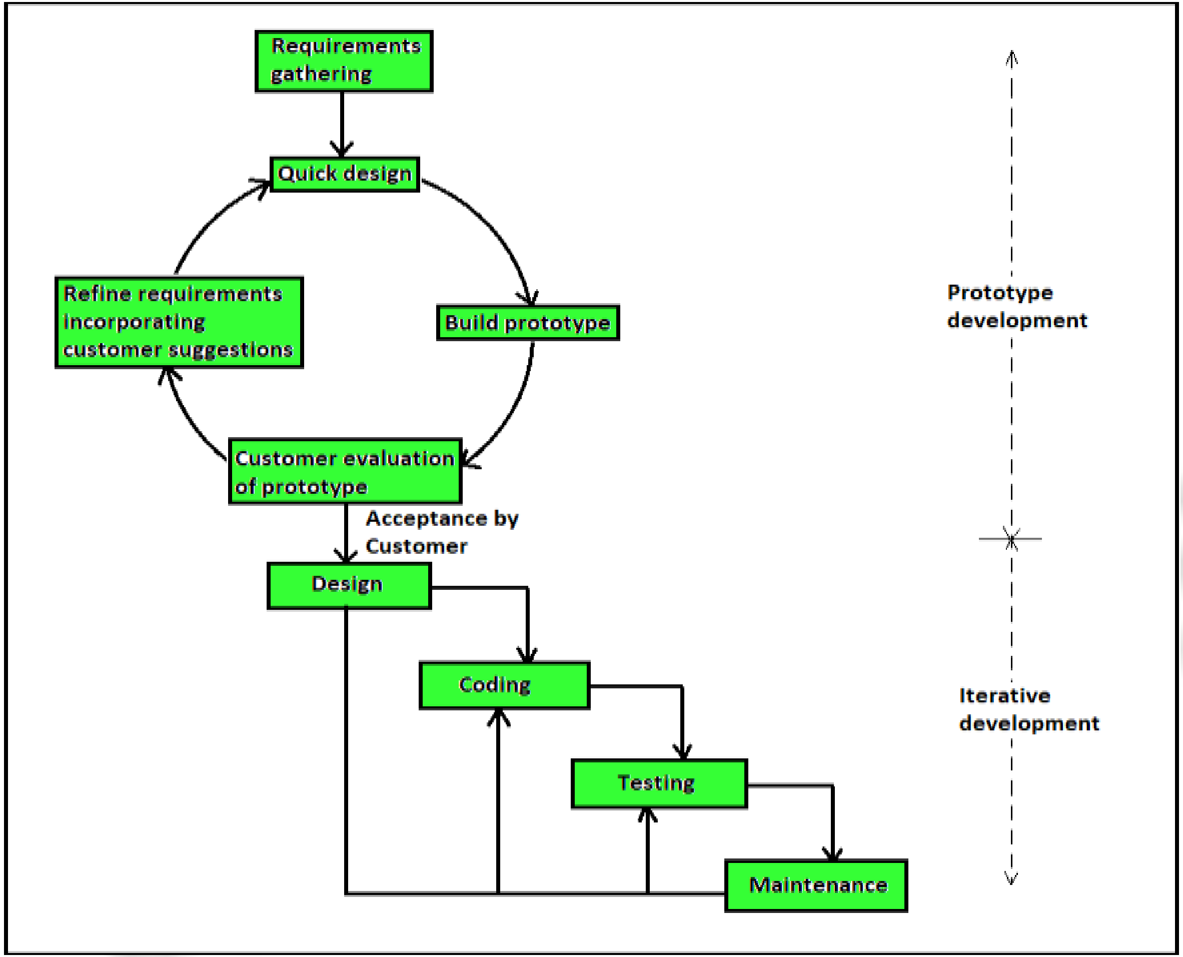
- V-shaped model is not good for ongoing projects.
- Requirement change at the later stage would cost too high.

3) Prototype Model

The prototype model is a model in which the prototype is developed prior to the actual software.

Prototype models have limited functional capabilities and inefficient performance when compared to the actual software. Dummy functions are used to create prototypes. This is a valuable mechanism for understanding the customers’ needs.

Software prototypes are built prior to the actual software to get valuable feedback from the customer. Feedbacks are implemented and the prototype is again reviewed by the customer for any change. This process goes on until the model is accepted by the customer.


Requirements gathering
Quick design
Build prototype
Customer evaluation of prototype
Refine requirements incorporating customer suggestions
Acceptance by Customer
Design
Coding
Testing
Maintenance
Prototype development
Iterative development

Advantages of Prototype Model:

•Prototype model reduces the cost and time of development as the defects are found much earlier.

•Missing feature or functionality or a change in requirement can be identified in the evaluation phase and can be implemented in the refined prototype.

•Involvement of a customer from the initial stage reduces any confusion in the requirement or understanding of any functionality.

Disadvantages of Prototype Model:

•Since the customer is involved in every phase, the customer can change the requirement of the end product which increases the complexity of the scope and may increase the delivery time of the product.

4) Spiral Model

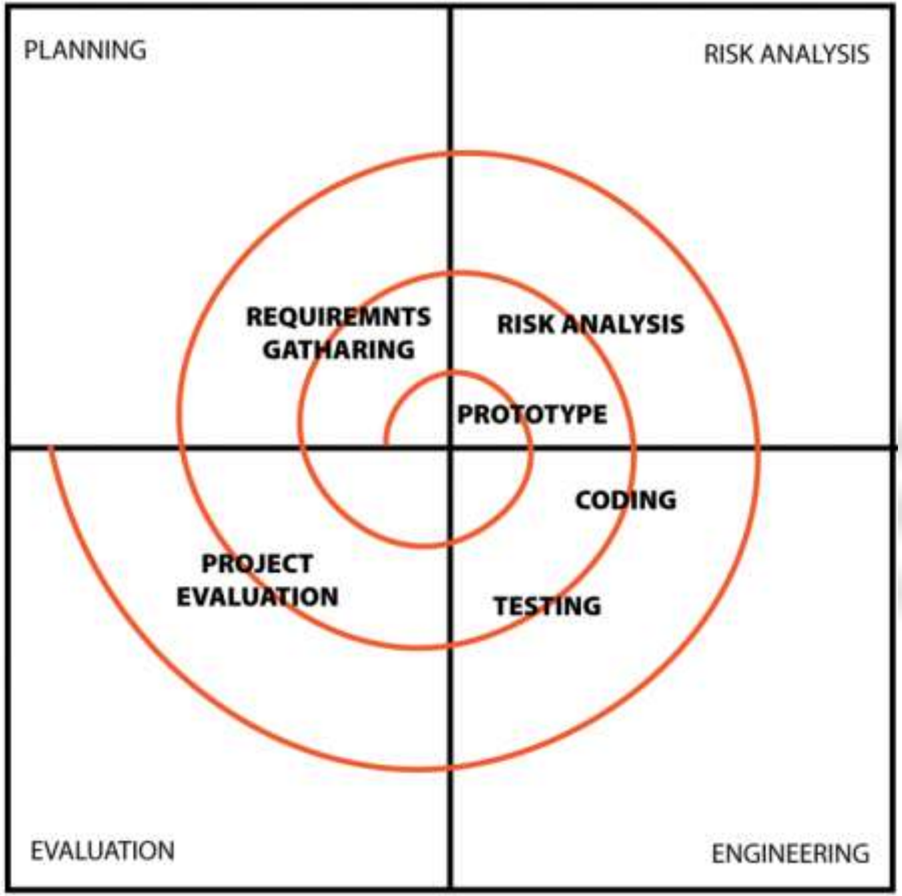
The Spiral Model includes iterative and prototype approach.

Spiral model phases are followed in the iterations.

The loops in the model represent the phase of the SDLC process i.e. the innermost loop is of requirement gathering & analysis which follows the Planning, Risk analysis, development, and evaluation. Next loop is Designing followed by Implementation & then testing.

Spiral Model has four phases:

- Planning
- Risk Analysis
- Engineering
- Evaluation


PLANNING
RISK ANALYSIS
REQUIREMNTS GATHARING
RISK ANALYSIS
PROTOTYPE
CODING
PROJECT EVALUATION
TESTING
EVALUATION
ENGINEERING

(i) Planning:

The planning phase includes requirement gathering wherein all the required information is gathered from the customer and is documented. Software requirement specification document is created for the next phase.

(ii) Risk Analysis:

In this phase, the best solution is selected for the risks involved and analysis is done by building the prototype.

(iii) Engineering:

Once the risk analysis is done, coding and testing are done.

(iv) Evaluation:

Customer evaluates the developed system and plans for the next iteration.

Advantages of Spiral Model:

- Risk Analysis is done extensively using the prototype models.
- Any enhancement or change in the functionality can be done in the next iteration.

Disadvantages of Spiral Model:

- The spiral model is best suited for large projects only.
- The cost can be high as it might take a large number of iterations which can lead to high time to reach the final product.

5) Iterative Incremental Model

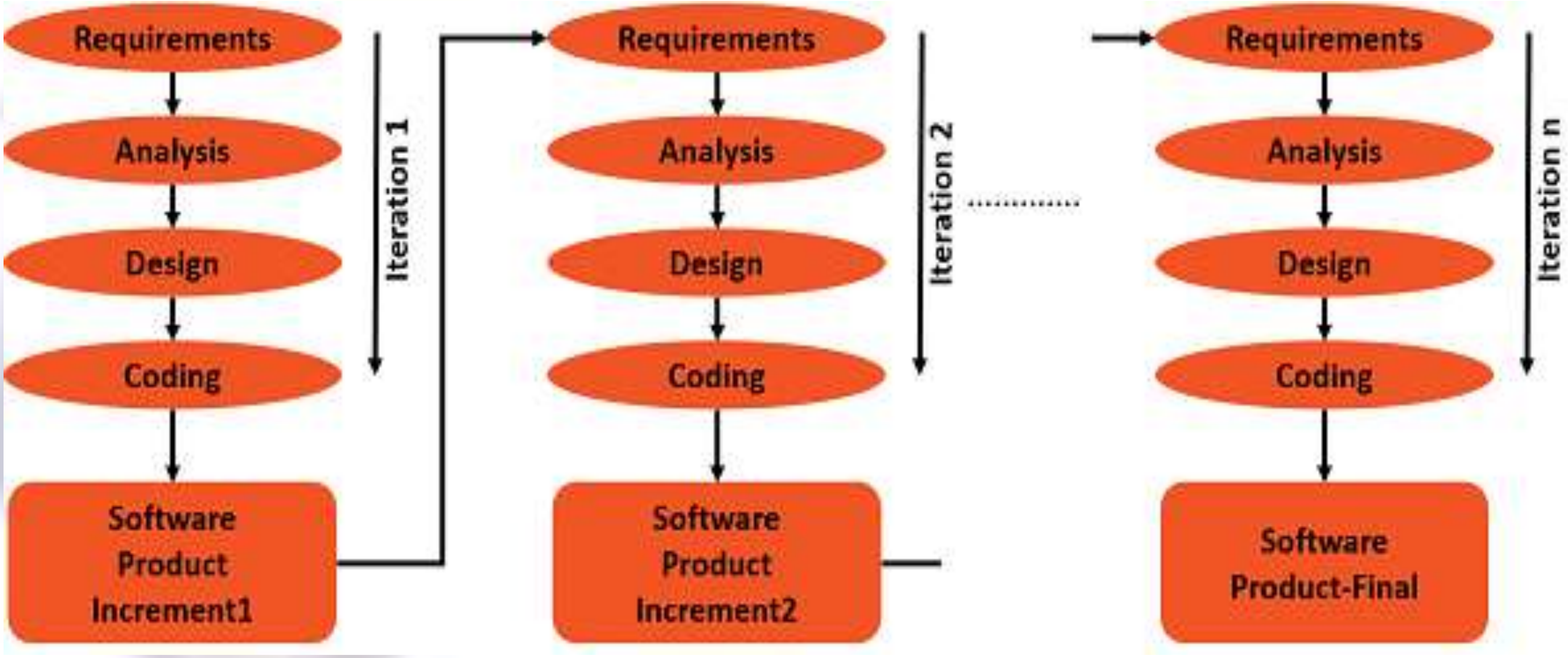
The iterative incremental model divides the product into small chunks.

For Example, Feature to be developed in the iteration is decided and implemented. Each iteration goes through the phases namely Requirement Analysis, Designing, Coding, and Testing. Detailed planning is not required in iterations.

Once the iteration is completed, a product is verified and is delivered to the customer for their evaluation and feedback.

Customer's feedback is implemented in the next iteration along with the newly added feature.

Hence, the product increments in terms of features and once the iterations are completed the final build holds all the features of the product.


Requirements
Analysis
Design
Coding
Software Product Increment1
Iteration 1
Requirements
Analysis
Design
Coding
Software Product Increment2
Iteration 2
Requirements
Analysis
Design
Coding
Software Product-Final
Iteration n

Phases of Iterative & Incremental Development Model:

- Inception phase
- Elaboration Phase
- Construction Phase
- Transition Phase

(i) Inception Phase:

Inception phase includes the requirement and scope of the Project.

(ii) Elaboration Phase:

In the elaboration phase, the working architecture of a product is delivered which covers the risk identified in the inception phase and also fulfills the non-functional requirements.

(iii) Construction Phase:

In the Construction phase, the architecture is filled in with the code which is ready to be deployed and is created through analysis, designing, implementation, and testing of the functional requirement.

(iv) Transition Phase:

In the Transition Phase, the product is deployed in the Production environment.

Advantages of Iterative & Incremental Model:

- Any change in the requirement can be easily done and would not cost as there is a scope of incorporating the new requirement in the next iteration.
- Risk is analyzed & identified in the iterations.
- Defects are detected at an early stage.
- As the product is divided into smaller chunks it is easy to manage the product.

Disadvantages of Iterative & Incremental Model:

- Complete requirement and understanding of a product are required to break down and build incrementally.

6) Big Bang Model

Big Bang Model does not have any defined process.

Money and efforts are put together as the input and output come as a developed product which might be or might not be the same as what the customer needs.

Big Bang Model does not require much planning and scheduling. The developer does the requirement analysis & coding and develops the product as per his understanding.

This model is used for small projects only. There is no testing team and no formal testing is done, and this could be a cause for the failure of the project.

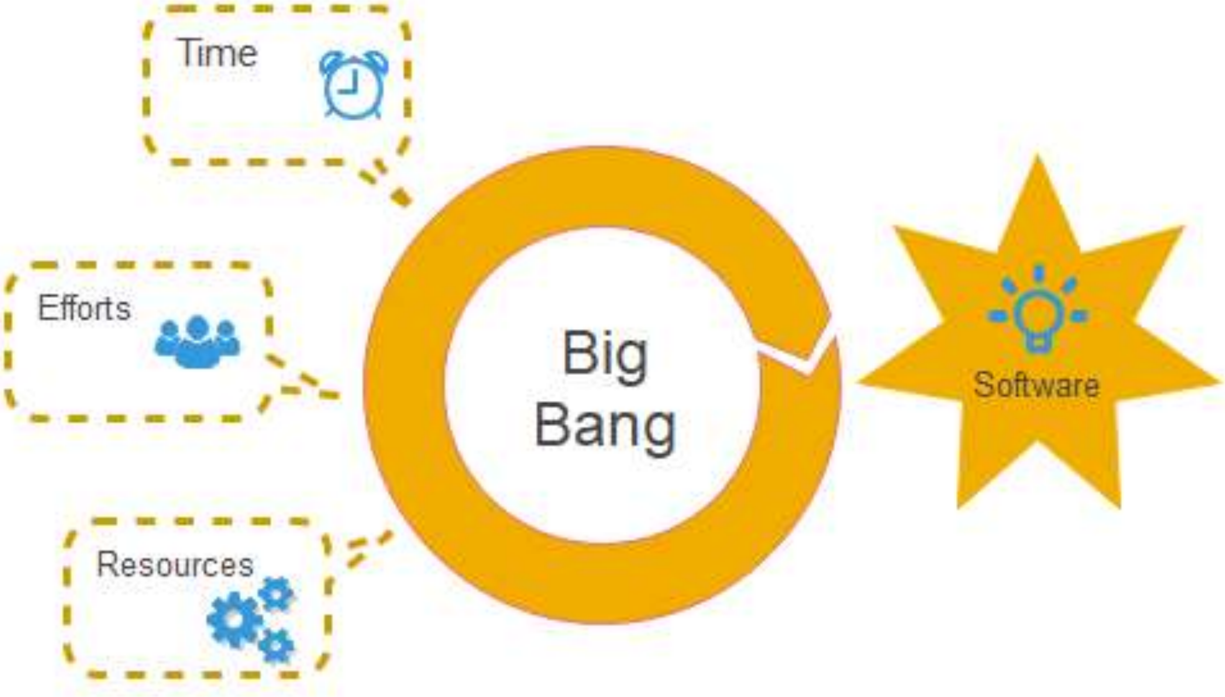


Fig. Big Bang Model

Advantages of Big Bang Model:

- It’s a very simple Model.
- Less Planning and scheduling is required.
- The developer has the flexibility to build the software of their own.

Disadvantages of the Big Bang Model:

- Big Bang models cannot be used for large, ongoing & complex projects.
- High risk and uncertainty.

7) Agile Model

Agile Model is a combination of the Iterative and incremental model.

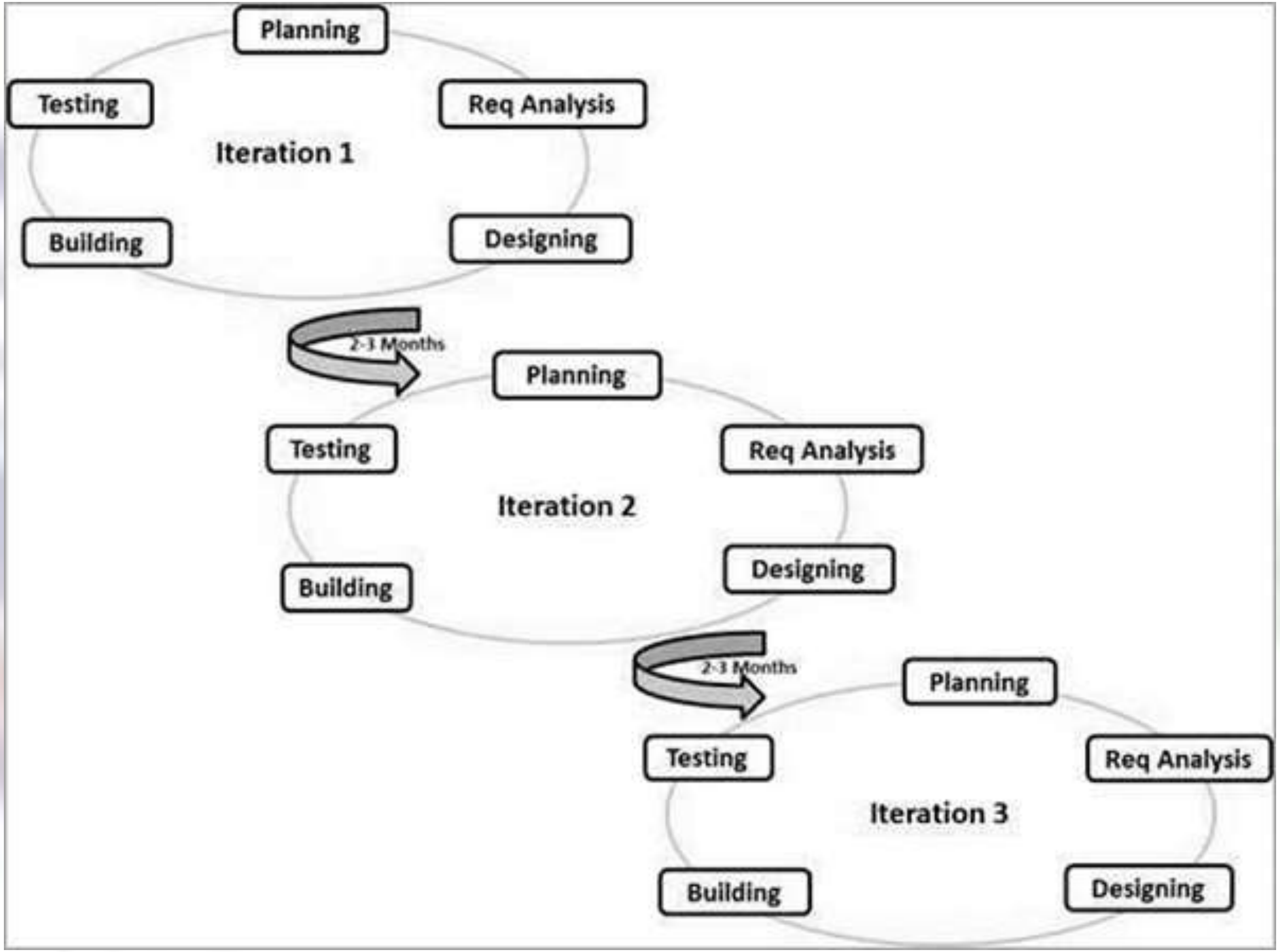
This model focuses more on flexibility while developing a product rather than on the requirement.

In Agile, a product is broken into small incremental builds.

It is not developed as a complete product in one go.

Each build increments in terms of features.

The next build is built on previous functionality.


Planning
Testing
Req Analysis
Iteration 1
Building
Designing
2-3 Months
Planning
Testing
Req Analysis
Iteration 2
Building
Designing
2-3 Months
Planning
Testing
Req Analysis
Iteration 3
Building
Designing

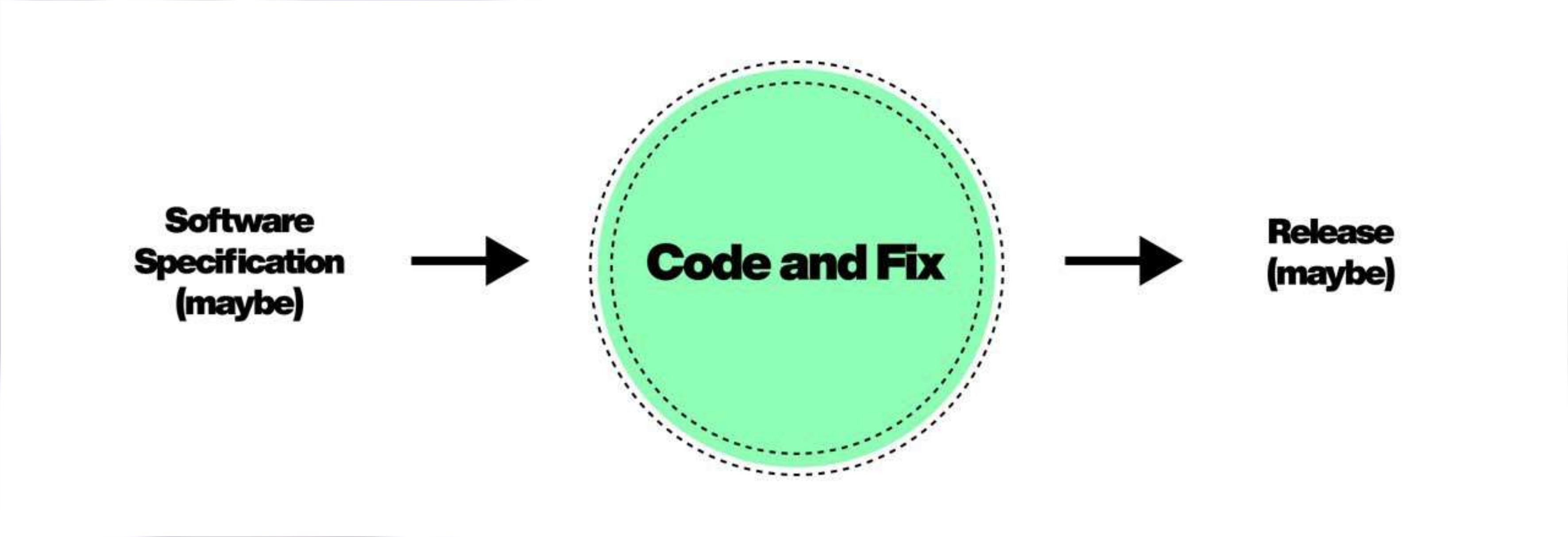
Advantages of Agile Model:

- It allows more flexibility to adapt to the changes.
- The new feature can be added easily.
- Customer satisfaction as the feedback and suggestions are taken at every stage.

Disadvantages:

- Lack of documentation.
- Agile needs experienced and highly skilled resources.
- If a customer is not clear about how exactly they want the product to be, then the project would fail.

8) Code and Fix Model



Conclusion

Adherence to a suitable life cycle is very important, for the successful completion of the Project. This, in turn, makes the management easier.

Different Software Development Life Cycle models have their own Pros and Cons.

The best model for any Project can be determined by the factors like Requirement (whether it is clear or unclear), System Complexity, Size of the Project, Cost, Skill limitation, etc.

Example, in case of an unclear requirement, Spiral and Agile models are best to be used as the required change can be accommodated easily at any stage.

Waterfall model is a basic model and all the other SDLC models are based on that only.

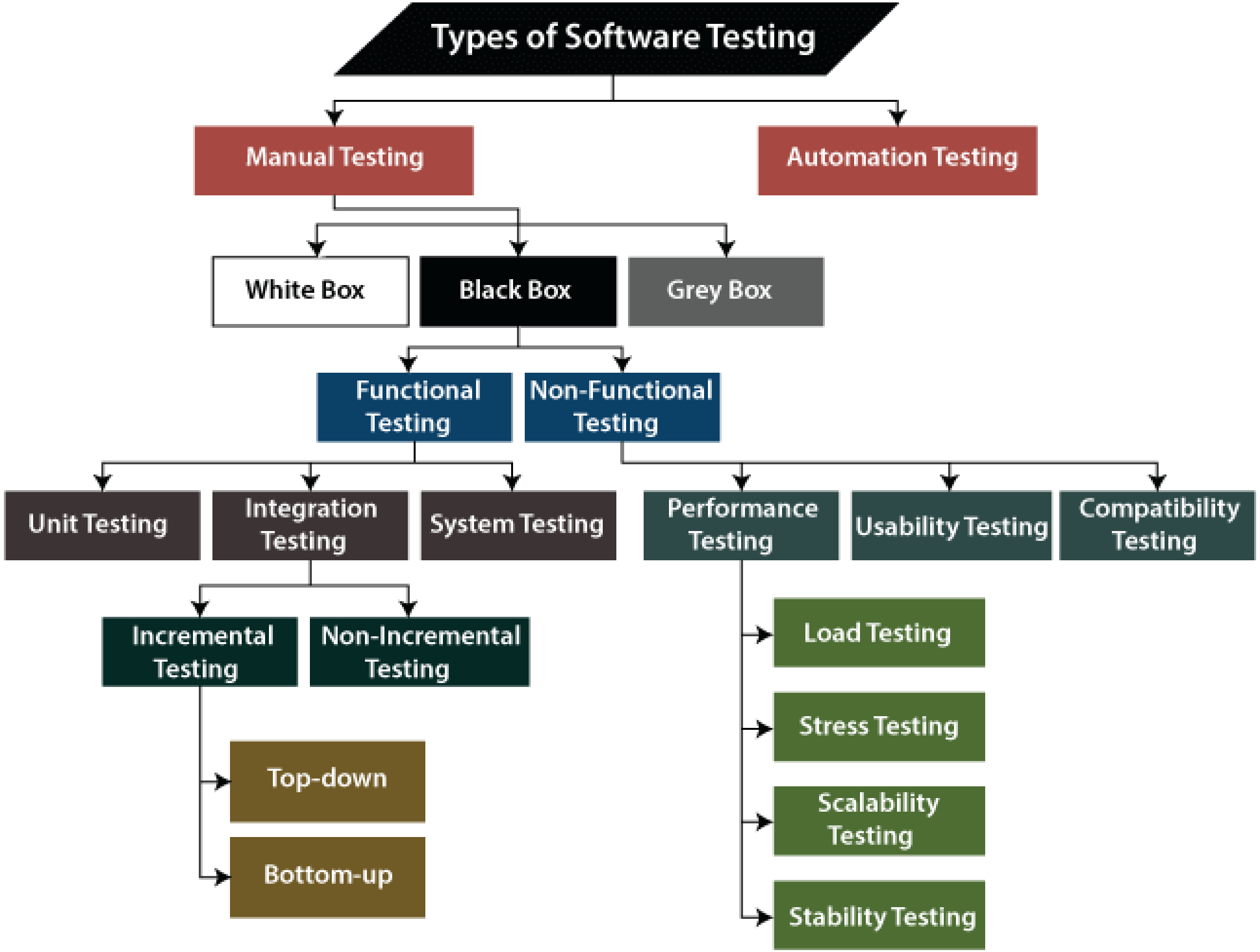
TESTING

What Is Software Testing?

Software testing is the process of finding errors in the developed product. It also checks whether the real outcomes can match expected results, as well as aids in the identification of defects, missing requirements, or gaps.

Importance of Software Testing?

- To deliver flawless product
- Enhance Product Quality
- Improve Security
- Detect Compatibility with different devices and platforms


Types of Software Testing
Manual Testing
Automation Testing
White Box
Black Box
Grey Box
Functional Testing
Non-Functional Testing
Unit Testing
Integration Testing
System Testing
Performance Testing
Usability Testing
Compatibility Testing
Incremental Testing
Non-Incremental Testing
Top-down
Bottom-up
Load Testing
Stress Testing
Scalability Testing
Stability Testing

The different types of functional testing include:

- **Unit testing.**

In unit testing, the tester checks individual software components. The aim is to test whether the components behave according to the requirements.

- **Integration testing.**

Integration testing deals with testing individual components or modules after they are combined in a group.

- **System testing.**

Here, the tester executes test cases for verifying the compliance of integrated and completed software along with specifications.

- **Sanity testing.** This tests logical reasoning related to the working of the program.

•**Smoke testing.**
Smoke testing tests simple and basic functionalities, such as if the user is able to log in or log out.

•**Interface testing.**
These tests check whether the communication between two software systems is correctly carried out.

•**Regression testing.**
This is probably one of the most important testing phases. Here, the old test cases of the entire application are executed after a new functionality has been implemented.

•**Beta/acceptance testing.**
Here, the intended users try the product and report bugs.

Non-functional testing types include:

- **Performance testing.** The performance or speed of the application is tested under the required workload.

- **Load testing.** This tests an application’s behavior under a huge workload. So, if you’re testing a website, load testing checks the site’s functionality and performance under high traffic.

- **Stress testing.** Stress testing determines software robustness by assessing whether it’s working beyond regular operation.

- **Volume testing.** This tests the performance of the system by loading the database to an increased volume of data.

- **Security testing.** Here, test cases are executed to check whether the system is safeguarded against sudden or deliberate attacks from internal and external sources.

- **Compatibility testing.** Test cases are executed to check whether the application is compatible with varying environments. For example, if you're testing a web application, compatibility testing deals with how the website works on different browsers or devices.

- **Install testing.** These tests check if a product works according to expectations after installation.

- **Recovery testing.** Here, testers determine an application's capacity to recover from hardware crashes and failures.

- **Reliability testing.** This procedure checks where an application can perform a particular task without failure within a specific timeframe.

Based on the amount of information you know about the product to test it, software testing can be divided into different types: **Black-box testing, White-box testing, and Grey-box testing.**

Black-box Testing

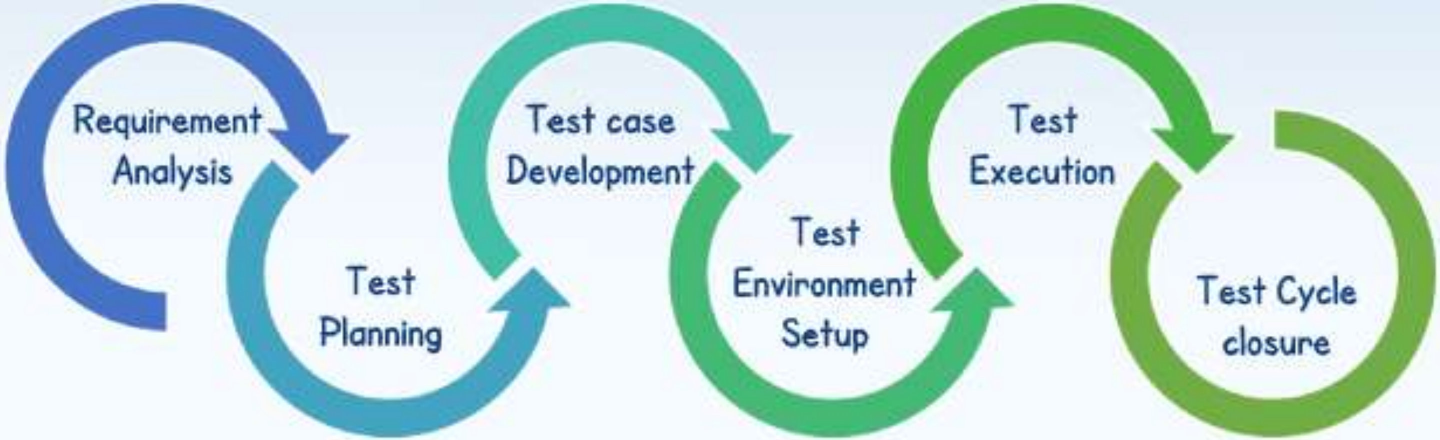
In this type of testing, you have the least amount of information on how the product is built. You don’t know about the structure of the product, its code, or logic.

In **white-box testing**, you have most of the information about the product. White-box testing is mostly used to make the code better.

Grey-box Testing

In this type of testing, you have partial information about the product. This type of testing is helpful to find out bugs that the user wouldn’t know about.

Phases of STLC Model



Requirement Analysis

Test Planning

Test case Development

Test Environment Setup

Test Execution

Test Cycle closure

Thank You!!!