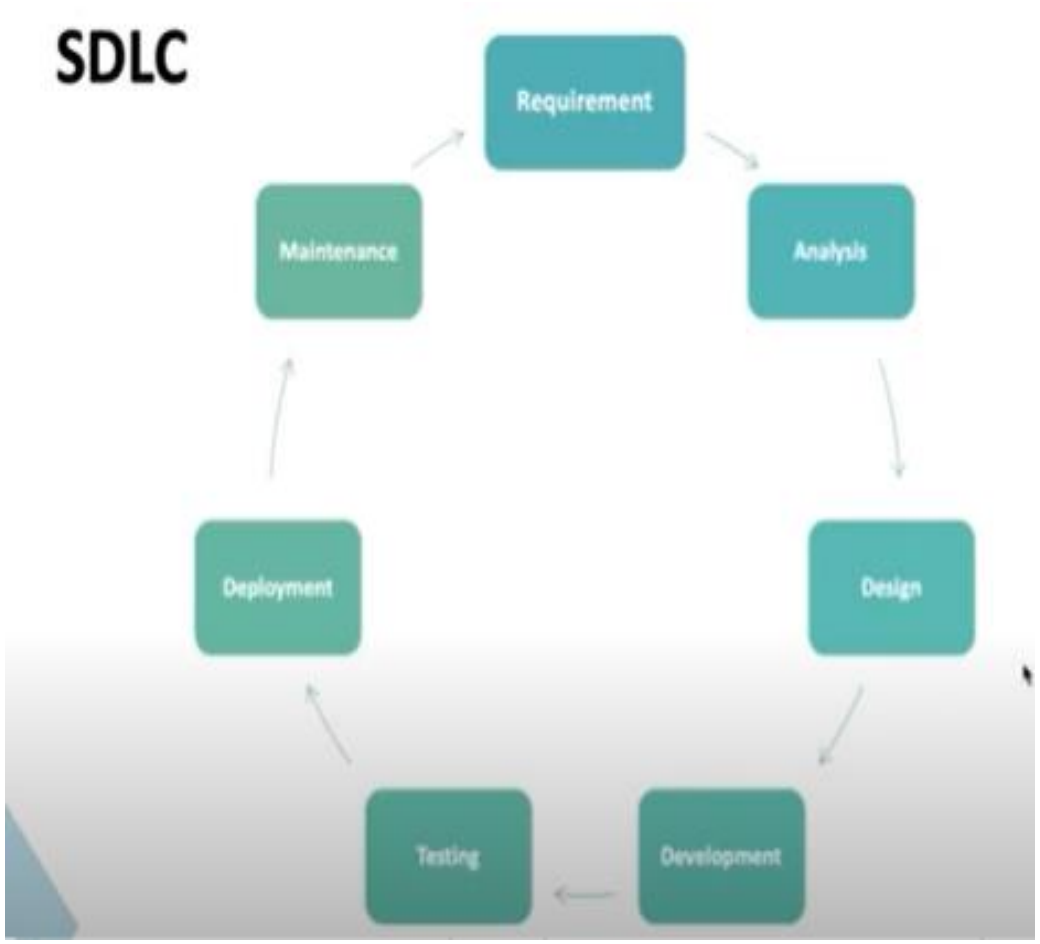
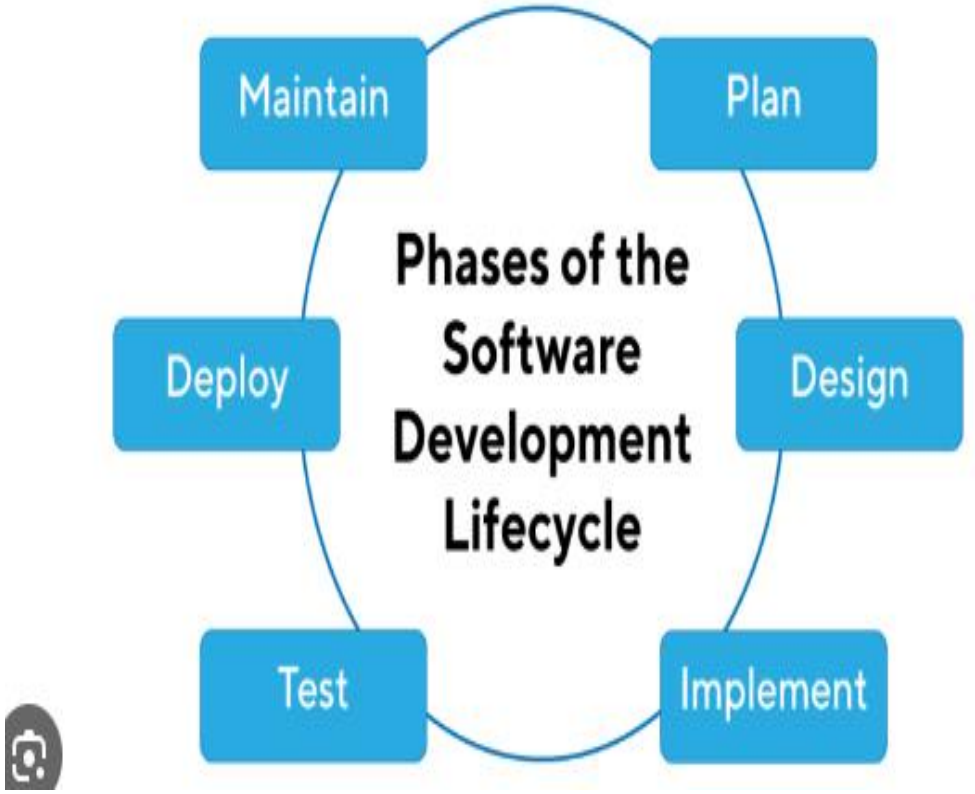
Software Engineering

- One of the earlier definitions of software engineering may be attributed to David Parnas, who stated that software engineering is a multiperson construction of multiversion software.
- Ian Sommerville states that ‘software engineering is an engineering discipline whose focus is the cost effective development of high quality software system’. He further explains that software is ‘abstract and intangible’ and defines software engineering as ‘an engineering discipline that is concerned with all aspect of software production from the early stages of the system specification to maintaining the system after it has gone into use’ This definition certainly captures the software life cycle from inception to project sunset.
- The software engineering 2004 Curriculum Guidelines for undergraduate degree programs in software engineering published by ACM and IEEE Computer society recognizes the broad scope of the field and highlights three definitions from different sources:

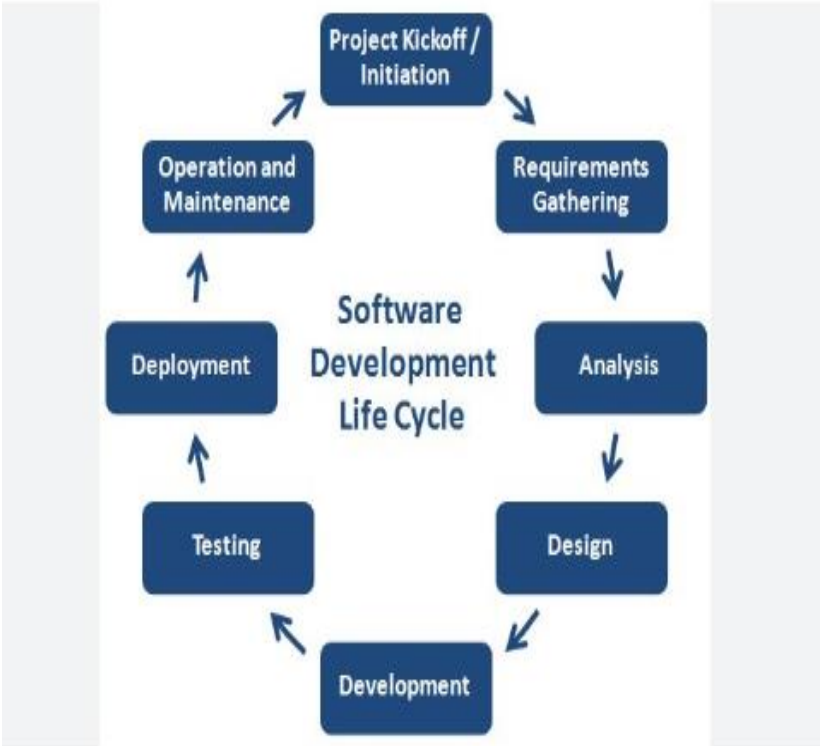
1. From F.L. Bauer: “The establishment and use of sound engineering principles (methods) in order to obtain economically software that is reliable and works on real machines.”
2. From CMU: “Software engineering is that form of engineering that applies the principles of computer science and mathematics to achieving cost effective solution to software problems.”
3. From IEEE: “ The application of a systematic, disciplined quantifiable approach to development, operation, and maintainance of software.”

Software Development Life Cycle






SDLC
(Software Development life Cycle)
Maintenance
Requirement Analysis
Defining
Designing
Coding
Testing
Deployment


Software Development Life Cycle
Project Kickoff / Initiation
Requirements Gathering
Analysis
Design
Development
Testing
Deployment
Operation and Maintenance

Need of SDLC

The development team must determine a suitable life cycle model for a particular plan and then observe to it.

Without using an exact life cycle model, the development of a software product would not be in a systematic and disciplined manner. When a team is developing a software product, there must be a clear understanding among team representative about when and what to do. Otherwise, it would point to chaos and project failure. This problem can be defined by using an example. Suppose a software development issue is divided into various parts and the parts are assigned to the team members. From then on, suppose the team representative is allowed the freedom to develop the roles assigned to them in whatever way they like. It is possible that one representative might start writing the code for his part, another might choose to prepare the test documents first, and some other engineer might begin with the design phase of the roles assigned to him. This would be one of the perfect methods for project failure.

A software life cycle model describes entry and exit criteria for each phase. A phase can begin only if its stage-entry criteria have been fulfilled. So without a software life cycle model, the entry and exit criteria for a stage cannot be recognized. Without software life cycle models, it becomes tough for software project managers to monitor the progress of the project.

A software life cycle model (also termed process model) is a pictorial and diagrammatic representation of the software life cycle. A life cycle model represents all the methods required to make a software product transit through its life cycle stages. It also captures the structure in which these methods are to be undertaken.

In other words, a life cycle model maps the various activities performed on a software product from its inception to retirement. Different life cycle models may plan the necessary development activities to phases in different ways. Thus, no element which life cycle model is followed, the essential activities are contained in all life cycle models though the action may be carried out in distinct orders in different life cycle models. During any life cycle stage, more than one activity may also be carried out.

The stages of SDLC are as follows:

Stage1: Planning and requirement analysis

Requirement Analysis is the most important and necessary stage in SDLC.

The senior members of the team perform it with inputs from all the stakeholders and domain experts or SMEs in the industry.

Planning for the quality assurance requirements and identifications of the risks associated with the projects is also done at this stage.

Business analyst and Project organizer set up a meeting with the client to gather all the data like what the customer wants to build, who will be the end user, what is the objective of the product. Before creating a product, a core understanding or knowledge of the product is very necessary.

For Example, A client wants to have an application which concerns money transactions. In this method, the requirement has to be precise like what kind of operations will be done, how it will be done, in which currency it will be done, etc.

Once the required function is done, an analysis is complete with auditing the feasibility of the growth of a product. In case of any ambiguity, a signal is set up for further discussion.

Once the requirement is understood, the SRS (Software Requirement Specification) document is created. The developers should thoroughly follow this document and also should be reviewed by the customer for future reference.

Stage2: Defining Requirements

Once the requirement analysis is done, the next stage is to certainly represent and document the software requirements and get them accepted from the project stakeholders.

This is accomplished through "SRS"- Software Requirement Specification document which contains all the product requirements to be constructed and developed during the project life cycle.

Stage3: Designing the Software

The next phase is about to bring down all the knowledge of requirements, analysis, and design of the software project. This phase is the product of the last two, like inputs from the customer and requirement gathering.

Stage4: Developing the project

In this phase of SDLC, the actual development begins, and the programming is built. The implementation of design begins concerning writing code. Developers have to follow the coding guidelines described by their management and programming tools like compilers, interpreters, debuggers, etc. are used to develop and implement the code.

Stage5: Testing

After the code is generated, it is tested against the requirements to make sure that the products are solving the needs addressed and gathered during the requirements stage.

During this stage, unit testing, integration testing, system testing, acceptance testing are done.

Stage6: Deployment

Once the software is certified, and no bugs or errors are stated, then it is deployed.

Then based on the assessment, the software may be released as it is or with suggested enhancement in the object segment.

After the software is deployed, then its maintenance begins.

Stage7: Maintenance

Once when the client starts using the developed systems, then the real issues come up and requirements to be solved from time to time.

This procedure where the care is taken for the developed product is known as maintenance.

Waterfall Model

1. The Waterfall model is the earliest SDLC approach that was used for software development.

2. Waterfall model is an example of a Sequential model. So it is also referred to as a **linear-sequential life cycle model**.

3. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin.

4. Also called as classic life cycle model.

The classical waterfall model is the basic software development life cycle model. It is very simple but idealistic. Earlier this model was very popular but nowadays it is not used. But it is very important because all the other software development life cycle models are based on the classical waterfall model.

Why Do We Use the Waterfall Model?

The waterfall model is a software development model used in the context of large, complex projects, typically in the field of information technology. It is characterized by a structured, sequential approach to project management and software development.

The waterfall model is useful in situations where the project requirements are well-defined and the project goals are clear. It is often used for large-scale projects with long timelines, where there is little room for error and the project stakeholders need to have a high level of confidence in the outcome.

Features of the Waterfall Model

1. **Sequential Approach**: The waterfall model involves a sequential approach to software development, where each phase of the project is completed before moving on to the next one.
2. **Document-Driven:** The waterfall model relies heavily on documentation to ensure that the project is well-defined and the project team is working towards a clear set of goals.
3. **Quality Control:** The waterfall model places a high emphasis on quality control and testing at each phase of the project, to ensure that the final product meets the requirements and expectations of the stakeholders.
4. **Rigorous Planning**: The waterfall model involves a rigorous planning process, where the project scope, timelines, and deliverables are carefully defined and monitored throughout the project lifecycle.

Overall, the waterfall model is used in situations where there is a need for a highly structured and systematic approach to software development. It can be effective in ensuring that large, complex projects are completed on time and within budget, with a high level of quality and customer satisfaction.

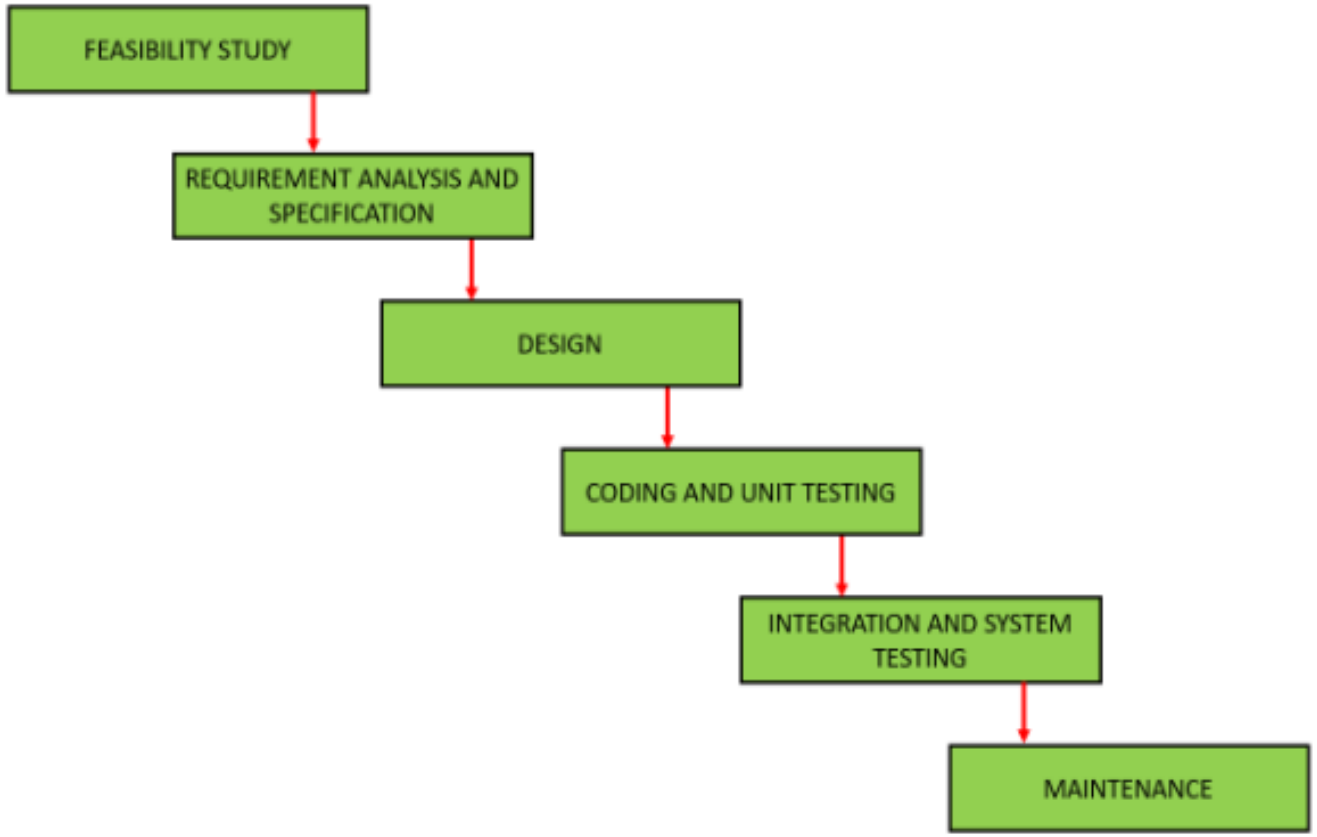
Phases of Classical Waterfall Model

Waterfall Model is a classical software development methodology that was first introduced by Winston W. Royce in 1970. It is a linear and sequential approach to software development that consists of several phases that must be completed in a specific order. The phases include:

1. **Requirements Gathering and Analysis:** The first phase involves gathering requirements from stakeholders and analyzing them to understand the scope and objectives of the project.
2. **Design:** Once the requirements are understood, the design phase begins. This involves creating a detailed design document that outlines the software architecture, user interface, and system components.
3. **Implementation:** The implementation phase involves coding the software based on the design specifications. This phase also includes unit testing to ensure that each component of the software is working as expected.
4. **Testing:** In the testing phase, the software is tested as a whole to ensure that it meets the requirements and is free from defects.

5. **Deployment:** Once the software has been tested and approved, it is deployed to the production environment.
6. **Maintenance:** The final phase of the Waterfall Model is maintenance, which involves fixing any issues that arise after the software has been deployed and ensuring that it continues to meet the requirements over time.

The classical waterfall model divides the life cycle into a set of phases. This model considers that one phase can be started after the completion of the previous phase. That is the output of one phase will be the input to the next phase. Thus the development process can be considered as a sequential flow in the waterfall. Here the phases do not overlap with each other. The different sequential phases of the classical waterfall model are shown in the below figure.



Phases of Classical Waterfall Model

Let us now learn about each of these phases in detail.

1. Feasibility Study

The main goal of this phase is to determine whether it would be financially and technically feasible to develop the software.
The feasibility study involves understanding the problem and then determining the various possible strategies to solve the problem. These different identified solutions are analyzed based on their benefits and drawbacks, The best solution is chosen and all the other phases are carried out as per this solution strategy.

2. Requirements Analysis and Specification

The aim of the requirement analysis and specification phase is to understand the exact requirements of the customer and document them properly. This phase consists of two different activities.

- **Requirement gathering and analysis:** Firstly all the requirements regarding the software are gathered from the customer and then the gathered requirements are analyzed. The goal of the analysis part is to remove incompleteness (an incomplete requirement is one in which some parts of the actual requirements have been omitted) and inconsistencies (an inconsistent requirement is one in which some part of the requirement contradicts some other part).
- **Requirement specification:** These analyzed requirements are documented in a software requirement specification (SRS) document. SRS document serves as a contract between the development team and customers. Any future dispute between the customers and the developers can be settled by examining the SRS document.

3. Design

The goal of this phase is to convert the requirements acquired in the SRS into a format that can be coded in a programming language. It includes high-level and detailed design as well as the overall software architecture. A Software Design Document is used to document all of this effort (SDD)

4. Coding and Unit Testing

In the coding phase software design is translated into source code using any suitable programming language. Thus each designed module is coded. The aim of the unit testing phase is to check whether each module is working properly or not.

5. Integration and System testing

Integration of different modules is undertaken soon after they have been coded and unit tested. Integration of various modules is carried out incrementally over a number of steps. During each integration step, previously planned modules are added to the partially integrated system and the resultant system is tested. Finally, after all the modules have been successfully integrated and tested, the full working system is obtained and system testing is carried out on this.
System testing consists of three different kinds of testing activities as described below.

- **Alpha testing:** Alpha testing is the system testing performed by the development team.
- **Beta testing:** Beta testing is the system testing performed by a friendly set of customers.
- **Acceptance testing:** After the software has been delivered, the customer performed acceptance testing to determine whether to accept the delivered software or reject it.

6. Maintenance

Maintenance is the most important phase of a software life cycle. The effort spent on maintenance is 60% of the total effort spent to develop a full software. There are basically three types of maintenance.

- **Corrective Maintenance:** This type of maintenance is carried out to correct errors that were not discovered during the product development phase.
- **Perfective Maintenance:** This type of maintenance is carried out to enhance the functionalities of the system based on the customer's request.
- **Adaptive Maintenance:** Adaptive maintenance is usually required for porting the software to work in a new environment such as working on a new computer platform or with a new operating system.

When Should You Use It?

1. Requirements are clear and fixed that may not change.
2. There are no ambiguous requirements.(no confusion)
3. It is good to use this model when the technology is well understood.
4. The project is short and cost is low.
5. Risk is zero or minimum.

Advantages:

1. Simple and easy to understand and use.
2. Easy to manage.
3. Works well for smaller and low budget projects where requirements are very well understood.
4. Clearly defined stages and Well understood.
5. Easy to arrange tasks.
6. Process and results are well documented.

Disadvantages:

1. No working software is produced until late during the life cycle.
2. High amounts of risk and uncertainty.
3. Not a good model for complex and object-oriented projects.
4. Poor model for long and ongoing projects.
5. It is difficult to measure progress within stages.
6. Cannot accommodate changing requirements.

Advantages of the Classical Waterfall Model

The classical waterfall model is an idealistic model for software development. It is very simple, so it can be considered the basis for other software development life cycle models. Below are some of the major advantages of this SDLC model.

- **Easy to Understand:** Classical Waterfall Model is very simple and easy to understand.
- **Individual Processing:** Phases in the Classical Waterfall model are processed one at a time.
- **Properly Defined:** In the classical waterfall model, each stage in the model is clearly defined.
- **Clear Milestones:** Classical Waterfall model has very clear and well-understood milestones.
- **Properly Documented:** Processes, actions, and results are very well documented.
- **Reinforces Good Habits:** Classical Waterfall Model reinforces good habits like define-before-design and design-before-code.
- **Working:** Classical Waterfall Model works well for smaller projects and projects where requirements are well understood.

Disadvantages of the Classical Waterfall Model

The Classical Waterfall Model suffers from various shortcomings, basically, we can't use it in real projects, but we use other software development lifecycle models which are based on the classical waterfall model. Below are some major drawbacks of this model.

- **No Feedback Path:** In the classical waterfall model evolution of software from one phase to another phase is like a waterfall. It assumes that no error is ever committed by developers during any phase. Therefore, it does not incorporate any mechanism for error correction.
- **Difficult to accommodate Change Requests:** This model assumes that all the customer requirements can be completely and correctly defined at the beginning of the project, but actually customer's requirements keep on changing with time. It is difficult to accommodate any change requests after the requirements specification phase is complete.
- **No Overlapping of Phases:** This model recommends that a new phase can start only after the completion of the previous phase. But in real projects, this can't be maintained. To increase efficiency and reduce cost, phases may overlap.

- **Limited Flexibility:** The Waterfall Model is a rigid and linear approach to software development, which means that it is not well-suited for projects with changing or uncertain requirements. Once a phase has been completed, it is difficult to make changes or go back to a previous phase.
- **Limited Stakeholder Involvement:** The Waterfall Model is a structured and sequential approach, which means that stakeholders are typically involved in the early phases of the project (requirements gathering and analysis) but may not be involved in the later phases (implementation, testing, and deployment).
- **Late Defect Detection:** In the Waterfall Model, testing is typically done toward the end of the development process. This means that defects may not be discovered until late in the development process, which can be expensive and time-consuming to fix.
- **Lengthy Development Cycle:** The Waterfall Model can result in a lengthy development cycle, as each phase must be completed before moving on to the next. This can result in delays and increased costs if requirements change or new issues arise.
- **Not Suitable for Complex Projects:** The Waterfall Model is not well-suited for complex projects, as the linear and sequential nature of the model can make it difficult to manage multiple dependencies and interrelated components.

Applications of Classical Waterfall Model

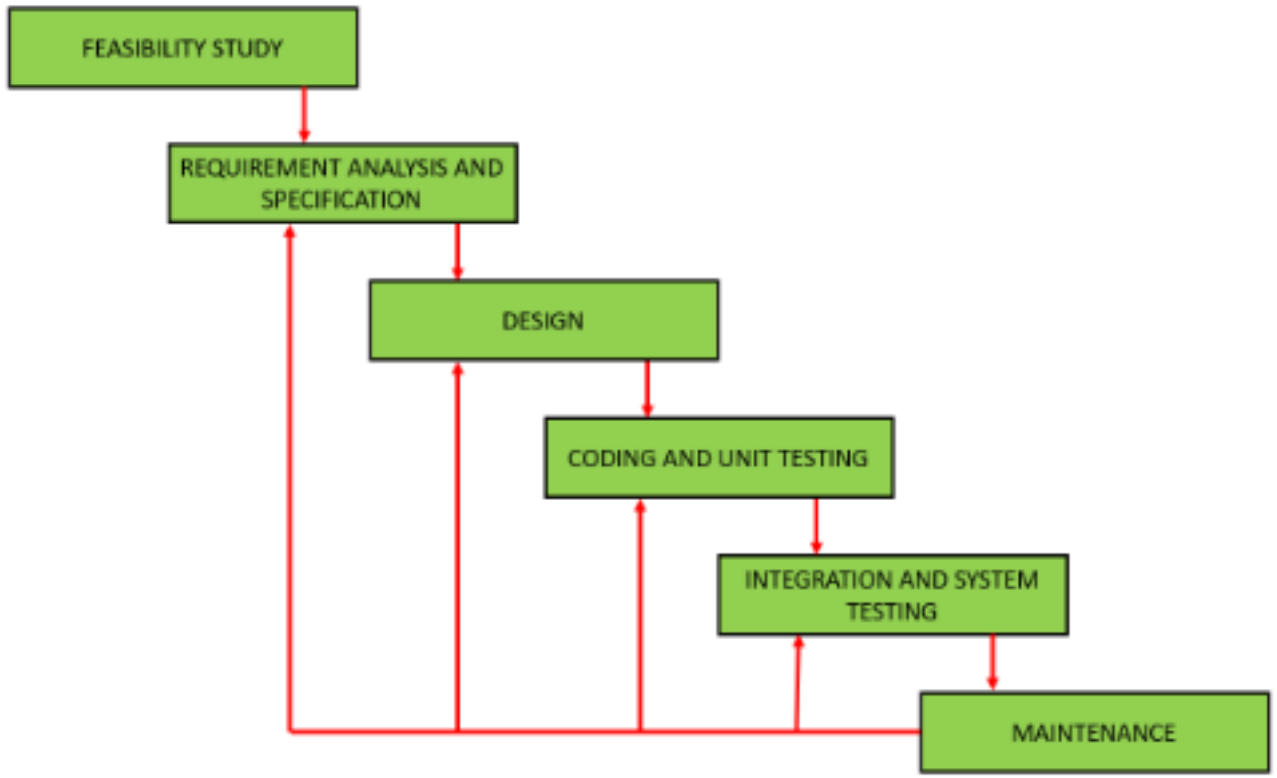
- **Large-scale Software Development Projects:** The Waterfall Model is often used for large-scale software development projects, where a structured and sequential approach is necessary to ensure that the project is completed on time and within budget.
- **Safety-Critical Systems:** The Waterfall Model is often used in the development of safety-critical systems, such as aerospace or medical systems, where the consequences of errors or defects can be severe.
- **Government and Defense Projects:** The Waterfall Model is also commonly used in government and defense projects, where a rigorous and structured approach is necessary to ensure that the project meets all requirements and is delivered on time.
- **Projects with well-defined Requirements:** The Waterfall Model is best suited for projects with well-defined requirements, as the sequential nature of the model requires a clear understanding of the project objectives and scope.
- **Projects with Stable Requirements:** The Waterfall Model is also well-suited for projects with stable requirements, as the linear nature of the model does not allow for changes to be made once a phase has been completed.

Iterative Waterfall Model

In a practical software development project, the classical waterfall model is hard to use. So, the Iterative waterfall model can be thought of as incorporating the necessary changes to the classical waterfall model to make it usable in practical software development projects. It is almost the same as the classical waterfall model except some changes are made to increase the efficiency of the software development.

The iterative waterfall model provides feedback paths from every phase to its preceding phases, which is the main difference from the classical waterfall model.

Feedback paths introduced by the iterative waterfall model are shown in the figure below.


FEASIBILITY STUDY
REQUIREMENT ANALYSIS AND SPECIFICATION
DESIGN
CODING AND UNIT TESTING
INTEGRATION AND SYSTEM TESTING
MAINTENANCE

When errors are detected at some later phase, these feedback paths allow for correcting errors committed by programmers during some phase. The feedback paths allow the phase to be reworked in which errors are committed and these changes are reflected in the later phases. But, there is no feedback path to the stage – feasibility study, because once a project has been taken, does not give up the project easily.

It is good to detect errors in the same phase in which they are committed. It reduces the effort and time required to correct the errors.

The Iterative Waterfall Model is a software development approach that combines the sequential steps of the traditional Waterfall Model with the flexibility of iterative design. It allows for improvements and changes to be made at each stage of the development process, instead of waiting until the end of the project.

Real-life example: Iterative Waterfall Model could be building a new website for a small business. The process might look like this:

Requirements gathering: This is the first stage where the business owners and developers meet to discuss the goals and requirements of the website.

Design: In this stage, the developers create a preliminary design of the website based on the requirements gathered in stage 1.

Implementation: In this stage, the developers begin to build the website based on the design created in stage 2.

Testing: Once the website has been built, it is tested to ensure that it meets the requirements and functions properly.

Deployment: The website is then deployed and made live to the public.

Review and improvement: After the website has been live for a while, the business owners and developers review its performance and make any necessary improvements.

This process is repeated until the website meets the needs and goals of the business. Each iteration builds upon the previous one, allowing for continuous improvement and iteration until the final product is complete.

Phase Containment of Errors: The principle of detecting errors as close to their points of commitment as possible is known as Phase containment of errors.

Collaboration: Throughout each stage of the process, there is collaboration between the business owners and developers. This ensures that the website meets the needs of the business and that any issues or concerns are addressed in a timely manner.

Flexibility: The iterative waterfall model allows for flexibility in the development process. If changes or new requirements arise, they can be incorporated into the next iteration of the website.

Testing and feedback: The testing stage of the process is important for identifying any issues or bugs that need to be addressed before the website is deployed. Additionally, feedback from users or customers can be gathered and used to improve the website in subsequent iterations.

Scalability: The iterative waterfall model is scalable, meaning it can be used for projects of various sizes and complexities. For example, a larger business may require more iterations or more complex requirements, but the same process can still be followed.

Maintenance: Once the website is live, ongoing maintenance is necessary to ensure it continues to meet the needs of the business and its users. The iterative waterfall model can be used for maintenance and improvement cycles, allowing the website to evolve and stay up-to-date.

Advantages of Iterative Waterfall Model :

- **Feedback Path –**
  In the classical waterfall model, there are no feedback paths, so there is no mechanism for error correction. But in the iterative waterfall model feedback path from one phase to its preceding phase allows correcting the errors that are committed and these changes are reflected in the later phases.
- **Simple –**
  Iterative waterfall model is very simple to understand and use. That's why it is one of the most widely used software development models.
- **Cost-Effective –**
  It is highly cost-effective to change the plan or requirements in the model. Moreover, it is best suited for agile organizations.
- **Well-organized –**
  In this model, less time is consumed on documenting and the team can spend more time on development and designing.
- **Risk Reduction:** The iterative approach allows for early identification and mitigation of risks, reducing the likelihood of costly errors later in the development process.

- **Quality Assurance:** The iterative approach promotes quality assurance by providing opportunities for testing and feedback throughout the development process. This results in a higher-quality end product.
- **Improved Customer Satisfaction:** The iterative approach allows for customer involvement and feedback throughout the development process, resulting in a final product that better meets the needs and expectations of the customer.
- **Predictable Outcomes:** The phased approach of the iterative waterfall model allows for more predictable outcomes and greater control over the development process, ensuring that the project stays on track and within budget.
- **Faster Time to Market:** The iterative approach allows for faster time to market as small and incremental improvements are made over time, rather than waiting for a complete product to be developed.
- **Easy to Manage:** The iterative waterfall model is easy to manage as each phase is well-defined and has a clear set of deliverables. This makes it easier to track progress, identify issues, and manage resources.

Drawbacks of Iterative Waterfall Model :

- **Difficult to incorporate change requests –**
  The major drawback of the iterative waterfall model is that all the requirements must be clearly stated before starting the development phase. Customers may change requirements after some time but the iterative waterfall model does not leave any scope to incorporate change requests that are made after the development phase starts.

- **Incremental delivery not supported –**
  In the iterative waterfall model, the full software is completely developed and tested before delivery to the customer. There is no scope for any intermediate delivery. So, customers have to wait a long for getting the software.

- **Overlapping of phases not supported –**
  Iterative waterfall model assumes that one phase can start after completion of the previous phase, But in real projects, phases may overlap to reduce the effort and time needed to complete the project.

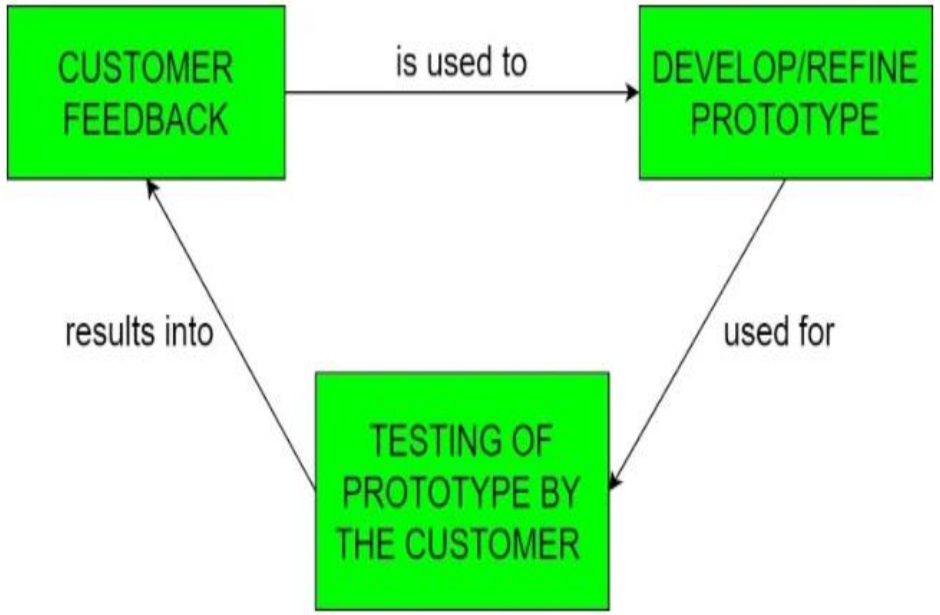
- **Risk handling not supported –**
  Projects may suffer from various types of risks. But, the Iterative waterfall model has no mechanism for risk handling.

- **Limited customer interactions –**
  Customer interaction occurs at the start of the project at the time of requirement gathering and at project completion at the time of software delivery. These fewer interactions with the customers may lead to many problems as the finally developed software may differ from the customers' actual requirements.

Prototyping Model

Prototyping is defined as the process of developing a working replication of a product or system that has to be engineered. It offers a small-scale facsimile of the end product and is used for obtaining customer feedback. The Prototyping concept is described below:

The Prototyping Model is one of the most popularly used Software Development Life Cycle Models (SDLC models). This model is used when the customers do not know the exact project requirements beforehand. In this model, a prototype of the end product is first developed, tested, and refined as per customer feedback repeatedly till a final acceptable prototype is achieved which forms the basis for developing the final product.



Prototyping Concept

In this process model, the system is partially implemented before or during the analysis phase thereby giving the customers an opportunity to see the product early in the life cycle. The process starts by interviewing the customers and developing the incomplete high-level paper model. This document is used to build the initial prototype supporting only the basic functionality as desired by the customer. Once the customer figures out the problems, the prototype is further refined to eliminate them. The process continues until the user approves the prototype and finds the working model to be satisfactory.

Steps Prototyping Model

Step 1: Requirement Gathering and Analysis: This is the initial step in designing a prototype model. In this phase, users are asked about what they expect or what they want from the system.

Step 2: Quick Design: This is the second step in Prototyping Model. This model covers the basic design of the requirement through which a quick overview can be easily described.

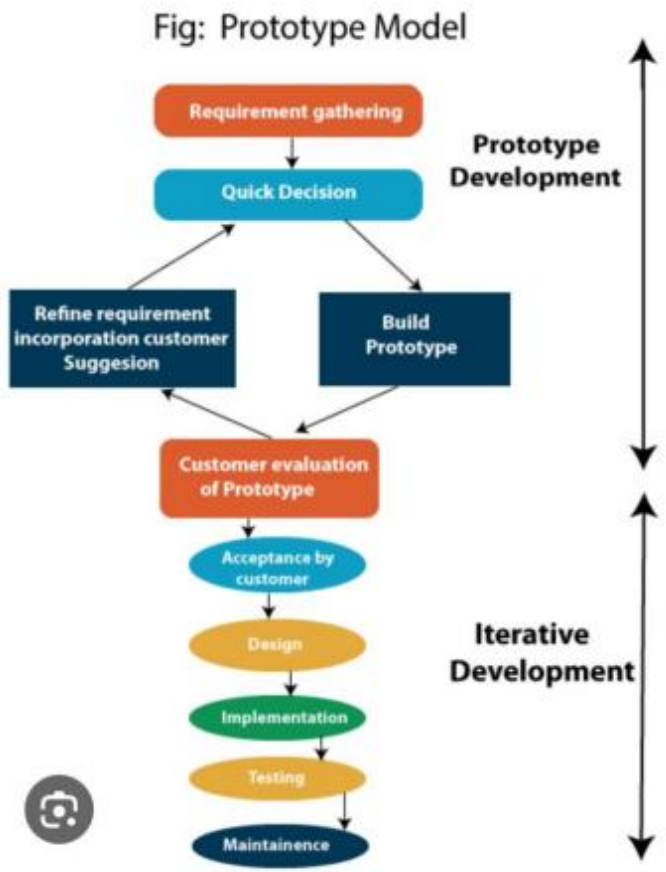
Step 3: Build a Prototype: This step helps in building an actual prototype from the knowledge gained from prototype design.

Step 4: Initial User Evaluation: This step describes the preliminary testing where the investigation of the performance model occurs, as the customer will tell the strength and weaknesses of the design, which was sent to the developer.

Step 5: Refining Prototype: If any feedback is given by the user, then improving the client's response to feedback and suggestions, the final system is approved.

Step 6: Implement Product and Maintain: This is the final step in the phase of the Prototyping Model where the final system is tested and distributed to production, here program is run regularly to prevent failures.

Prototype Model



Advantages of Prototyping Model

- The customers get to see the partial product early in the life cycle. This ensures a greater level of customer satisfaction and comfort.
- New requirements can be easily accommodated as there is scope for refinement.
- Missing functionalities can be easily figured out.
- Errors can be detected much earlier thereby saving a lot of effort and cost, besides enhancing the quality of the software.
- The developed prototype can be reused by the developer for more complicated projects in the future.
- Flexibility in design.
- Early feedback from customers and stakeholders can help guide the development process and ensure that the final product meets their needs and expectations.

- Prototyping can be used to test and validate design decisions, allowing for adjustments to be made before significant resources are invested in development.
- Prototyping can help reduce the risk of project failure by identifying potential issues and addressing them early in the process.
- Prototyping can facilitate communication and collaboration among team members and stakeholders, improving overall project efficiency and effectiveness.
- Prototyping can help bridge the gap between technical and non-technical stakeholders by providing a tangible representation of the product.

Disadvantages of the Prototyping Model

- Costly with respect to time as well as money.
- There may be too much variation in requirements each time the prototype is evaluated by the customer.
- Poor Documentation due to continuously changing customer requirements.
- It is very difficult for developers to accommodate all the changes demanded by the customer.
- There is uncertainty in determining the number of iterations that would be required before the prototype is finally accepted by the customer.
- After seeing an early prototype, the customers sometimes demand the actual product to be delivered soon.
- Developers in a hurry to build prototypes may end up with sub-optimal solutions.
- The customer might lose interest in the product if he/she is not satisfied with the initial prototype.
- The prototype may not be scalable to meet the future needs of the customer.
- The prototype may not accurately represent the final product due to limited functionality or incomplete features.
- The focus on prototype development may shift the focus away from the final product, leading to delays in the development process.
- The prototype may give a false sense of completion, leading to the premature release of the product.
- The prototype may not consider technical feasibility and scalability issues that can arise during the final product development.
- The prototype may be developed using different tools and technologies, leading to additional training and maintenance costs.
- The prototype may not reflect the actual business requirements of the customer, leading to dissatisfaction with the final product.

Applications of Prototyping Model

- The Prototyping Model should be used when the requirements of the product are not clearly understood or are unstable.
- The prototyping model can also be used if requirements are changing quickly.
- This model can be successfully used for developing user interfaces, high-technology software-intensive systems, and systems with complex algorithms and interfaces.
- The prototyping Model is also a very good choice to demonstrate the technical feasibility of the product.

Incremental Model

Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle. In this model, each module goes through the requirements, design, implementation and testing phases. Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieved.

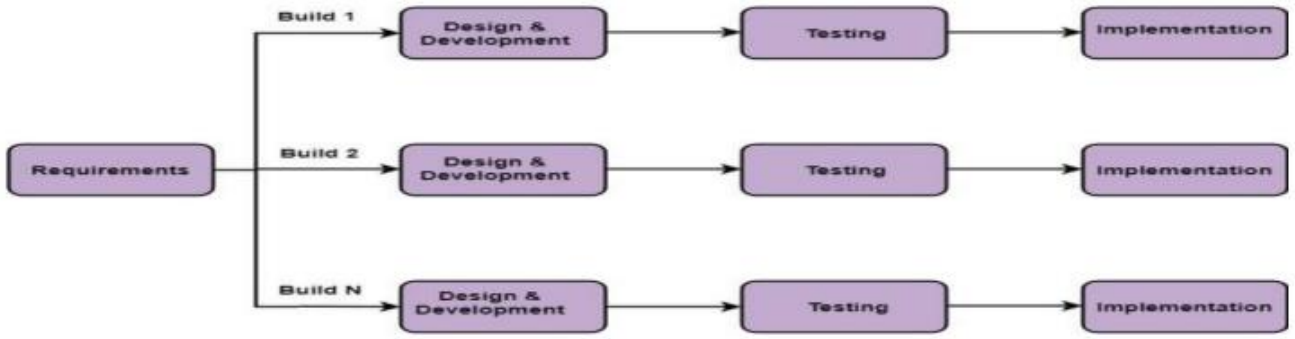


Fig: Incremental Model

Various phases of incremental Model

1. Requirement analysis: In the first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement analysis team. To develop the software under the incremental model, this phase performs a crucial role.

2. Design & Development: In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style and development phase.

3. Testing: In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behavior of each task.

4. Implementation: Implementation phase enables the coding phase of the development system. It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product

When we use the incremental model

- When the requirements are superior.
- A project has a lengthy development schedule.
- When Software team are not very well skilled or trained.
- When the customer demands a quick release of the product.
- You can develop prioritized requirements first.

Advantages

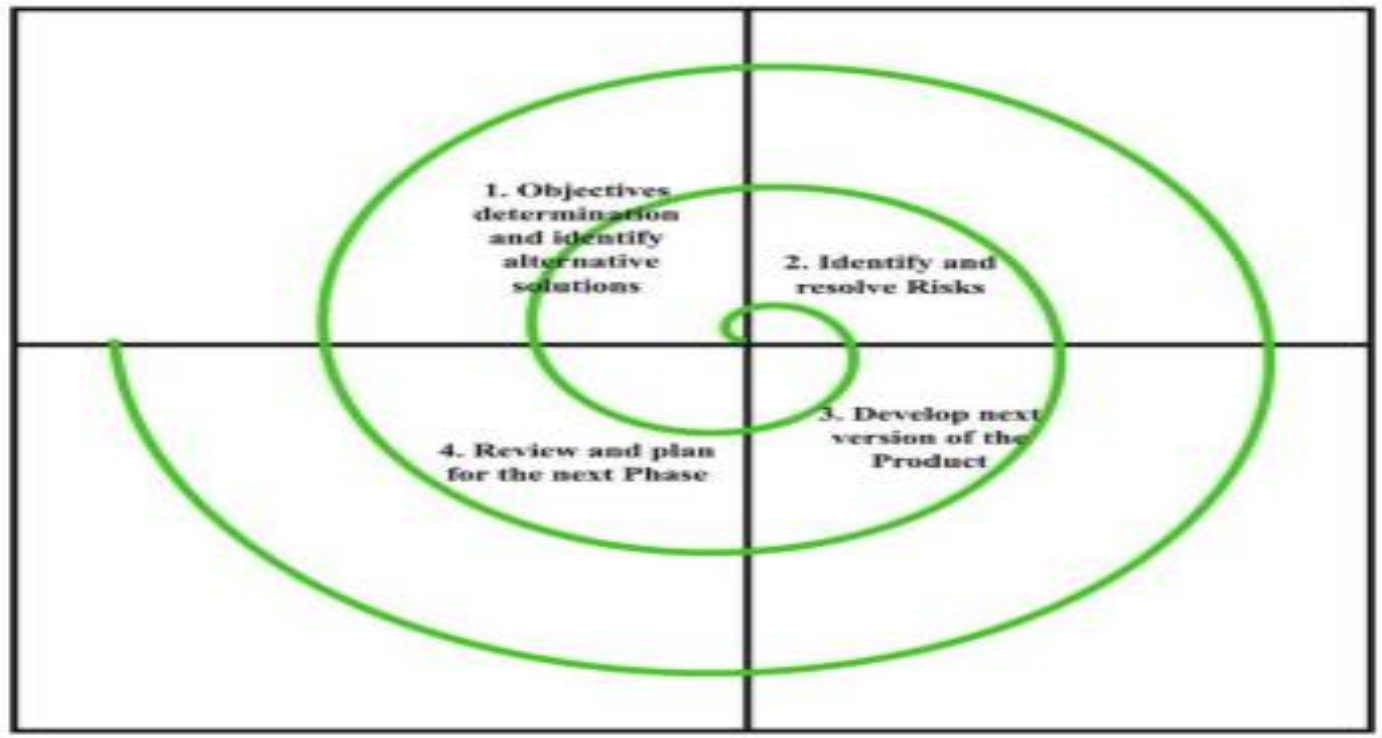
- Errors are easy to be recognized.
- Easier to test and debug
- More flexible.
- Simple to manage risk because it handled during its iteration.
- The Client gets important functionality early.

Disadvantages

- Need for good planning
- Total Cost is high.
- Well defined module interfaces are needed.

Spiral model

The below diagram shows the different phases of the Spiral Model: –



Each phase of the Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are discussed below-

1. **Objectives determination and identify alternative solutions:** Requirements are gathered from the customers and the objectives are identified, elaborated, and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.
2. **Identify and resolve Risks:** During the second quadrant, all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the Prototype is built for the best possible solution.
3. **Develop next version of the Product:** During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.
4. **Review and plan for the next Phase:** In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next phase is started.

Why Spiral Model is called Meta Model?

The Spiral model is called a Meta-Model because it subsumes all the other SDLC models. For example, a single loop spiral actually represents the Iterative Waterfall Model. The spiral model incorporates the stepwise approach of the Classical Waterfall Model. The spiral model uses the approach of the Prototyping Model by building a prototype at the start of each phase as a risk-handling technique. Also, the spiral model can be considered as supporting the Evolutionary model — the iterations along the spiral can be considered as evolutionary levels through which the complete system is built.

Advantages of Spiral Model:

Below are some advantages of the Spiral Model.

1. **Risk Handling:** The projects with many unknown risks that occur as the development proceeds, in that case, Spiral Model is the best development model to follow due to the risk analysis and risk handling at every phase.
2. **Good for large projects:** It is recommended to use the Spiral Model in large and complex projects.
3. **Flexibility in Requirements:** Change requests in the Requirements at later phase can be incorporated accurately by using this model.
4. **Customer Satisfaction:** Customer can see the development of the product at the early phase of the software development and thus, they habituated with the system by using it before completion of the total product.

Disadvantages of Spiral Model:

Below are some main disadvantages of the spiral model.

1. **Complex:** The Spiral Model is much more complex than other SDLC models.
2. **Expensive:** Spiral Model is not suitable for small projects as it is expensive.
3. **Too much dependability on Risk Analysis:** The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced experts, it is going to be a failure to develop a project using this model.
4. **Difficulty in time management:** As the number of phases is unknown at the start of the project, so time estimation is very difficult.
5. Complexity: The Spiral Model can be complex, as it involves multiple iterations of the software development process.

Evolutionary Model

What is Evolutionary Model?

Evolutionary model is also referred to as the **successive versions model** and sometimes as the **incremental model**. In Evolutionary model, the software requirement is first broken down into several modules (or functional units) that can be incrementally constructed and delivered (see Figure 5).

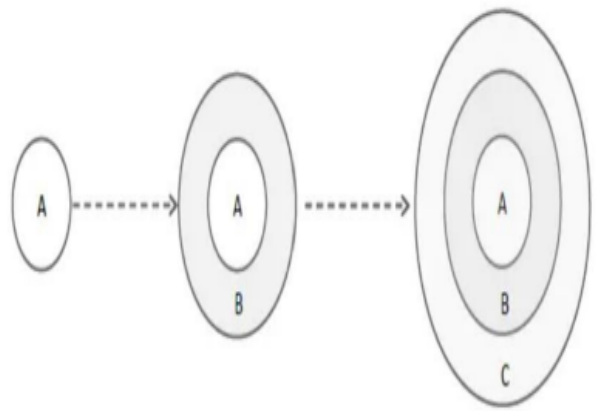


Figure 5: Evolutionary Development of a Software Product

Evolutionary Development of a Software Product

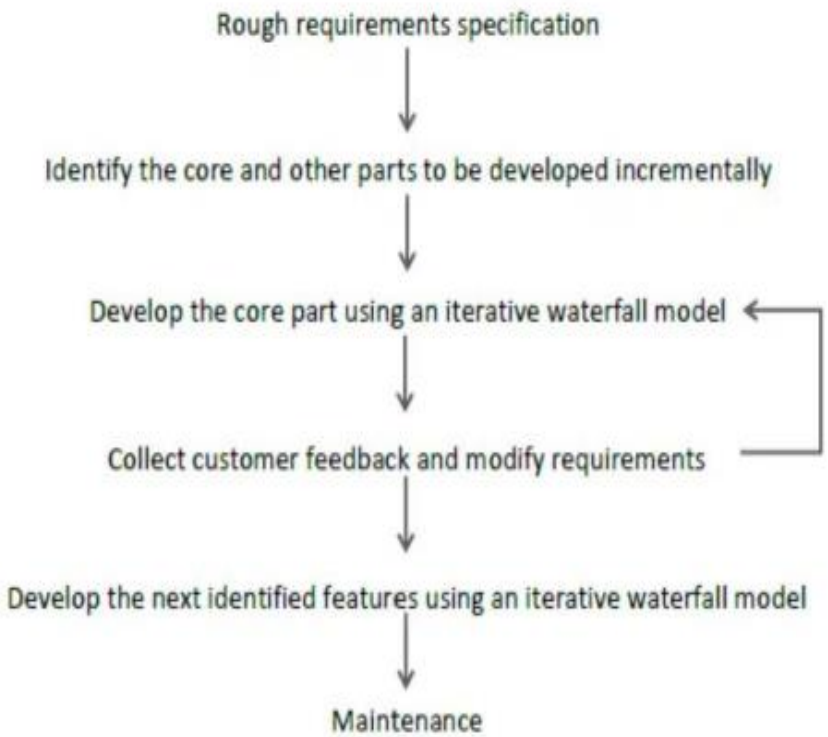


Figure 6: Evolutionary Model of Software Development

Evolutionary Model of Software Development

Each successive version/model of the product is a fully functioning software capable of performing more work than the previous versions/model.

The **evolutionary model** is normally useful for very large products, where it is easier to find modules for incremental implementation.

Often, **evolutionary model** is used when the customer prefers to receive the product in increments so that he can start using the different features as and when they are developed rather than waiting all the time for the full product to be developed and delivered.

Advantages of Evolutionary Model

- **Large project:** Evolutionary model is normally useful for very large products.
- User gets a **chance to experiment with a partially developed software** much before the complete version of the system is released.
- Evolutionary model helps to accurately **elicit user requirements** during the delivery of different versions of the software.
- The core modules get tested thoroughly, thereby **reducing the chances of errors** in the core modules of the final products.
- Evolutionary model **avoids the need to commit large resources** in one go for development of the system.

Disadvantages of Evolutionary Model

- Difficult to divide the problem into several versions that would be acceptable to the customer and which can be incrementally implemented and delivered.

Agile Methodologies

➢ **About Agile:**

- Mostly used model in todays digital era.
- Agile means "The ability to respond to changes from requirements, technology & people"
- It is an incremental and iterative process of software development.

➢ **Working with Example:**

- Divides requirements into multiple iterations & provide specific functionality for the release.
- Delivers multiple software requirements.
- Each iterations are lasts from two to three weeks.
- Direct collaboration with customers.
- Rapid project development.

WHAT IS AGILE?

Traditional method vs Agile Model

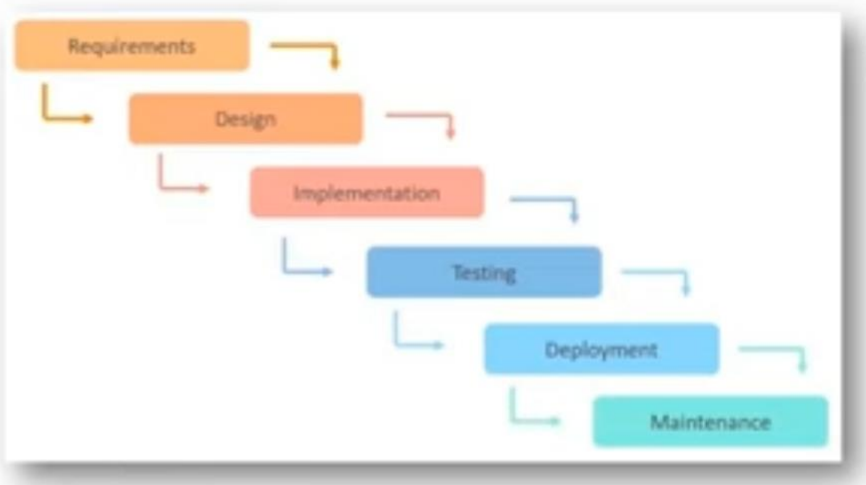
For Example:

Instagram Social Application:

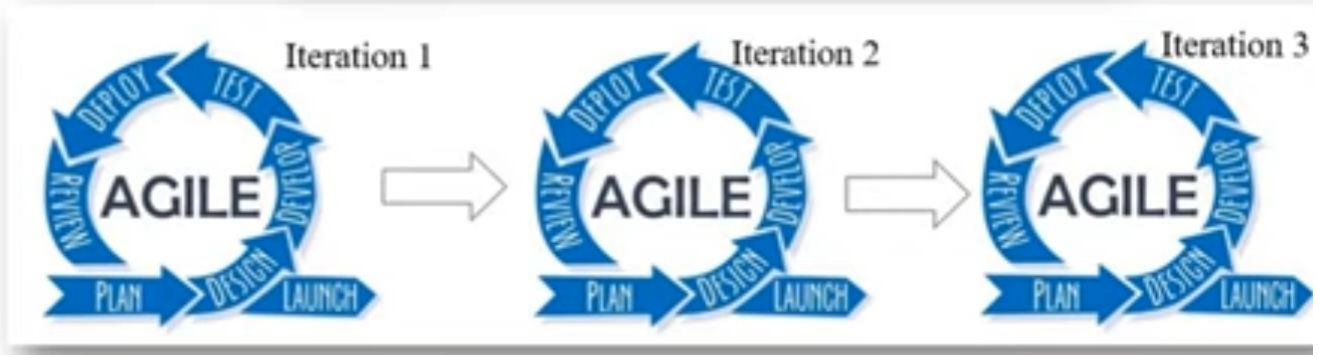
Requirements are:

1. Follow – Unfollow Option
2. Edit profile
3. Search
4. Messaging
5. Post photos
6. Upload story
7. To make reels
8. Go Live

Waterfall Model



Agile Model

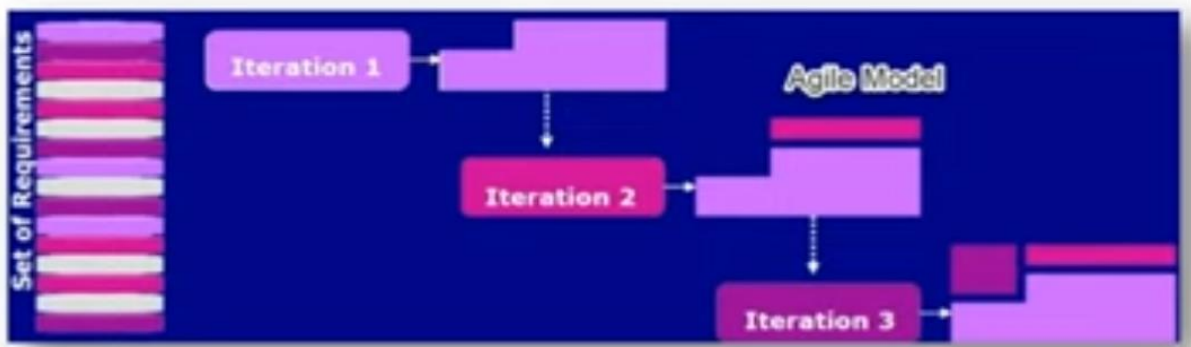


Agile principles

1. Highest priority is to satisfy the customers to early & continue delivery of software.
2. Being flexible about changing requirements at any point of development.
3. Working on frequent and short deliveries like couple of weeks or months with preference.
4. Transparency between business people and developers and requires them to work together.
5. By providing a better productive environment and providing them with all the support, motivation it leads to better productivity.
6. Face to face communication as the most effective way to communicate between customer & development team.
7. Continuous attention towards effective designing and technical excellence through following optimal code standard.

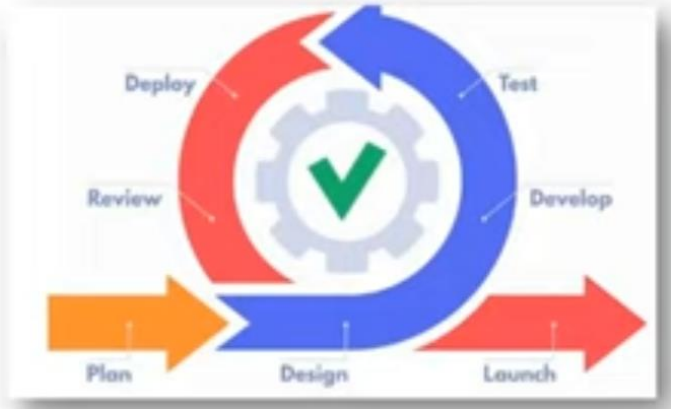
Advantages of Agile Model

1. Supports customer involvement and customer satisfaction
2. Strong communication of the software team with the customer.
3. Little planning required
4. Efficient design and fulfils the business requirement.
5. Anytime changes are acceptable.
6. Provides a very realistic approach to software development
7. Updated versions of functioning software are released every week.
8. It reduces total development time.

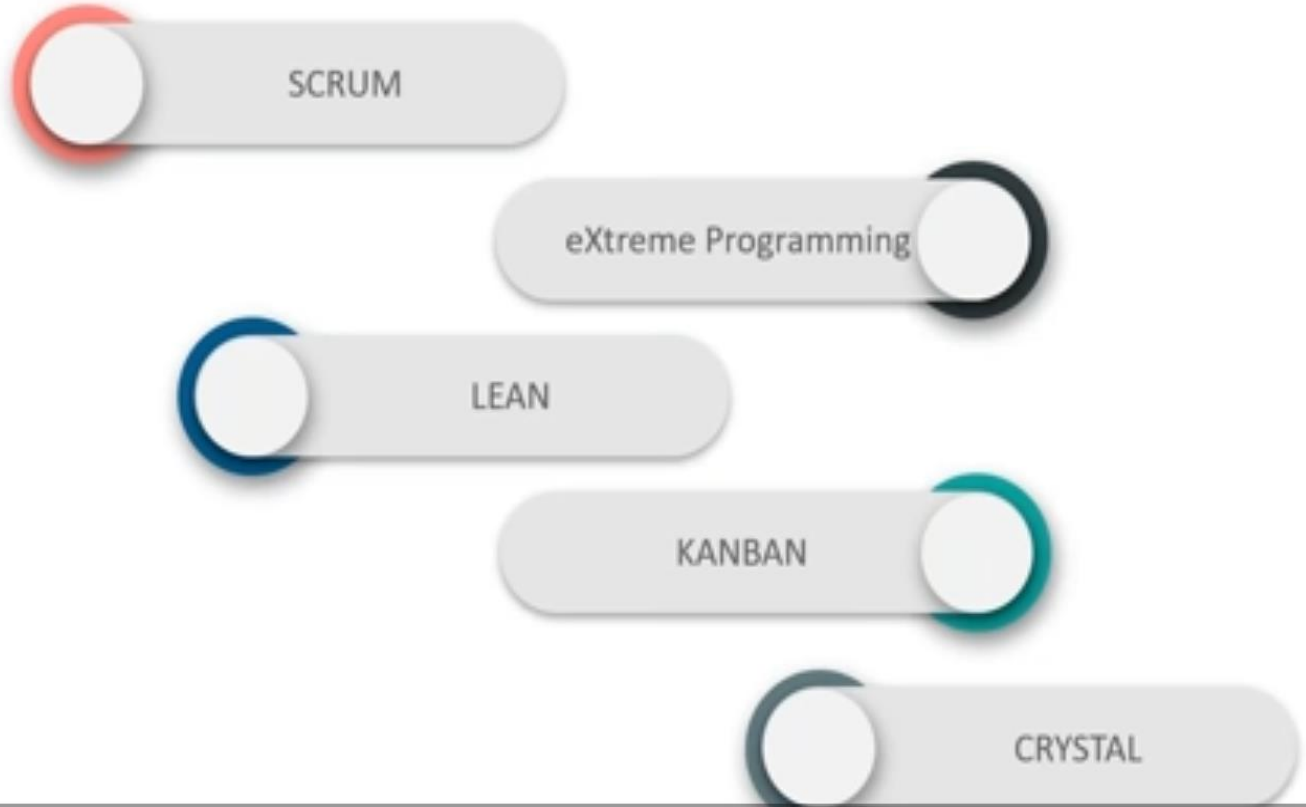


Disadvantages of Agile

1. Due to the lack of proper documentation, once the project completes and the developers allotted to another project, maintenance of the finished project can become a difficulty.
2. Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.

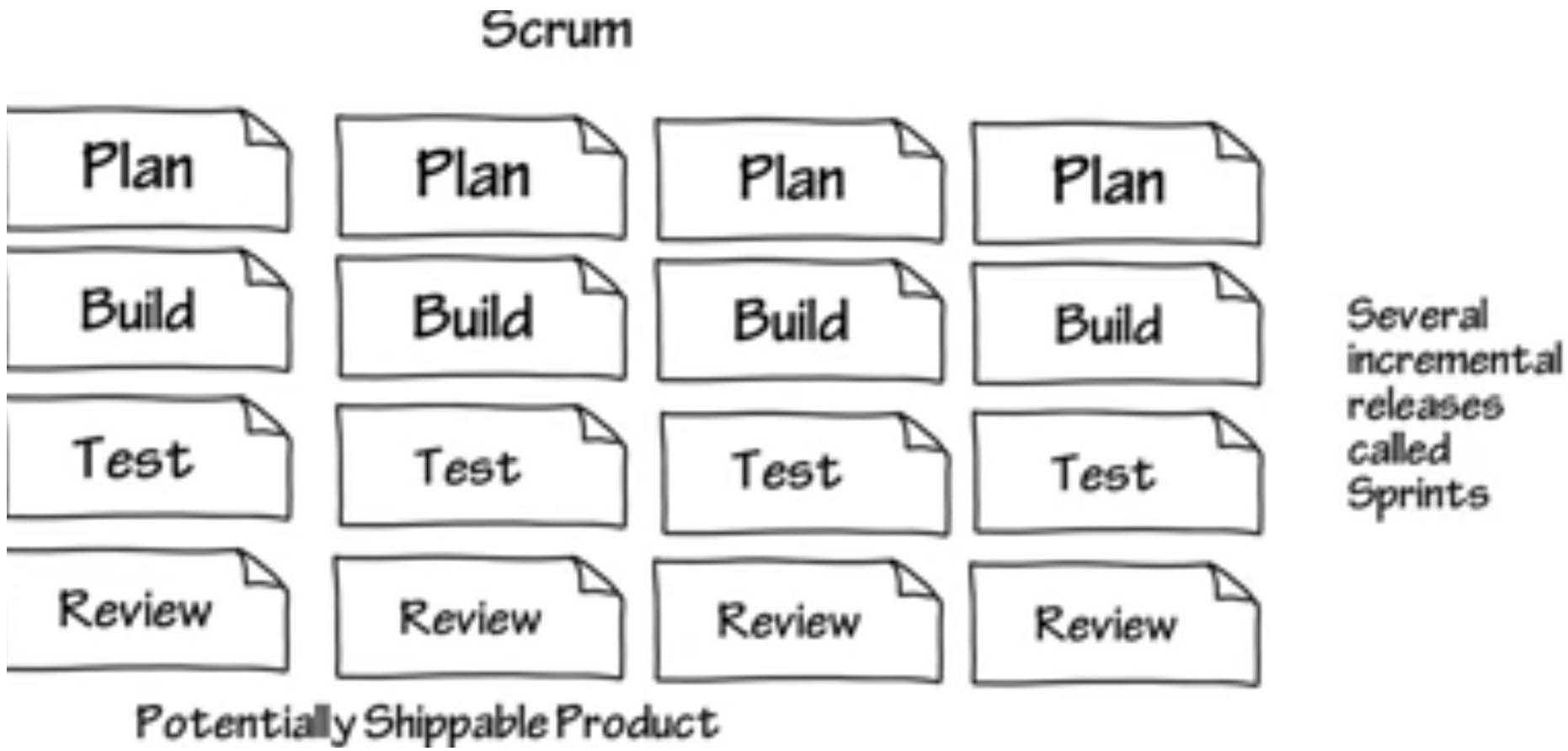


HOW TO IMPLEMENT AGILE?



Scrum

- One of the most popular agile methodology
- Scrum is lightweight, iterative and incremental framework
- Scrum breakdown the development phases into stages or cycles called sprint.
- The development time for each sprint is maximized and dedicated thereby managing only one sprint at a time.
- Scrum team has scrum master and product owner with constant communication on a daily basis.
- Keywords: Backlog, sprint, daily scrum, Scrum master, product owner.


Scrum
Plan
Plan
Plan
Plan
Build
Build
Build
Build
Test
Test
Test
Test
Review
Review
Review
Review
Several incremental releases called Sprints
Potentially Shippable Product

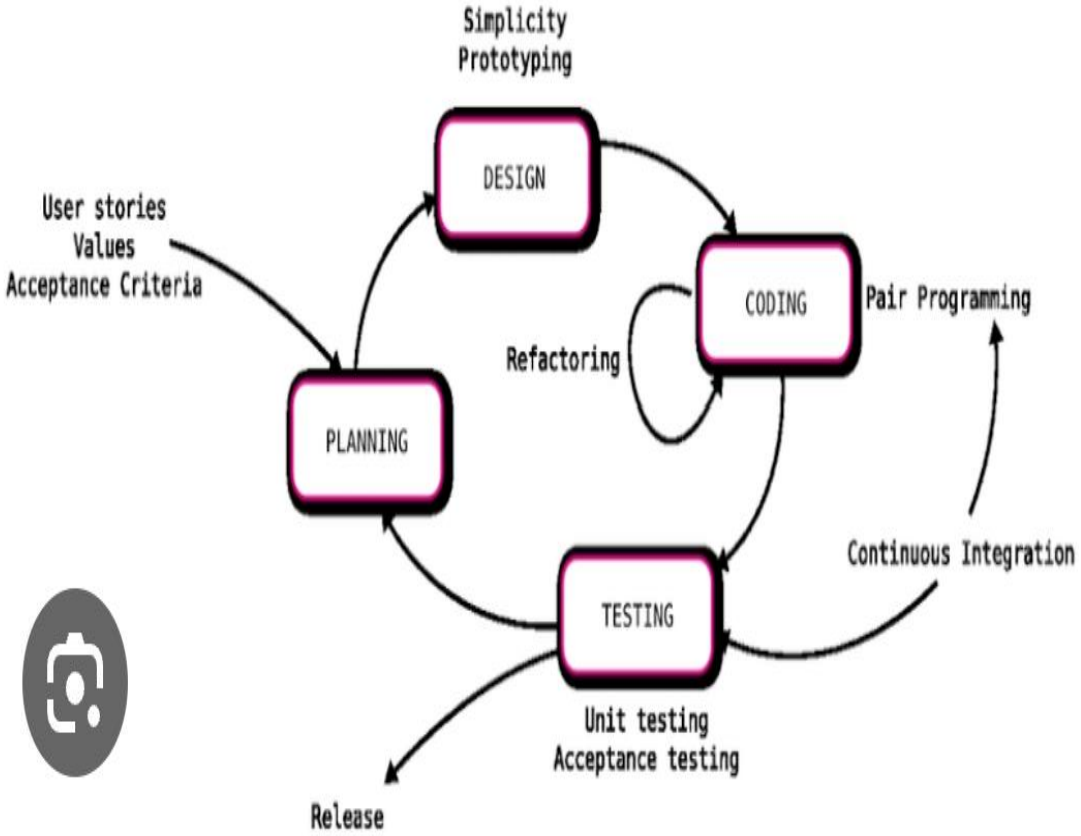
Advantages

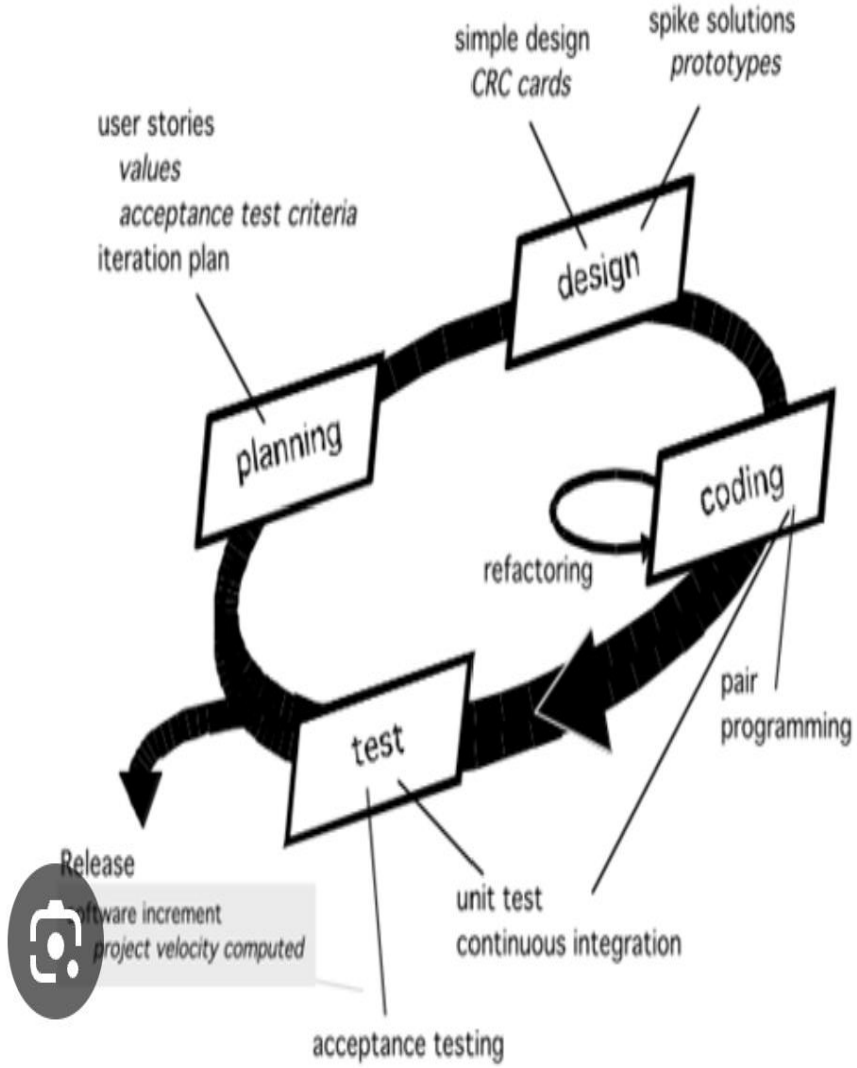
- Freedom and adaption
- High quality and low risk product
- Reduce the development time upto 40%
- Scrum customer satisfaction is very important
- Reviewing the current sprint before moving to new one

Disadvantages

- More efficient for small team size.
- No changes in the sprint

Extreme programming





Extreme Programming (XP) is an Agile software development methodology that focuses on delivering high-quality software through frequent and continuous feedback, collaboration, and adaptation. XP emphasizes a close working relationship between the development team, the customer, and stakeholders, with an emphasis on rapid, iterative development and deployment.

- Agile development approaches evolved in the 1990s as a reaction to documentation and bureaucracy based approaches particularly the waterfall approach. Agile approaches are based on some common principles ,some of which are

1. Working software is the key measure of progress in a project.
2. For progress in a project, therefore software should be developed and delivered rapidly in small increments.
3. Even late changes in the requirements should be entertained.
4. Face-to-face communication is preferred over documentation.

5. Continuous feedback and involvement of customer is necessary for developing good-quality software.
6. Simple design which envolves and improves with time is a better approach than doing an elaborate design up front for handling all possible scenarios.
7. The delivery dates are decided by empowered teams of talented individuals.

What are Extreme Programming values?

XP initially recognized 5 values. By the way, the fifth value was added in the second edition. Here they are:

Communication

Some people compare software development with team sports activities where participants rely on each other and transfer knowledge from one to another. Extreme Programming highlights the importance of such communication using whiteboards and face to face discussion.

Simplicity

Simplicity is connected with the simplest thing that will work.

It's crucial to do only the necessary things. Simplicity also means requirements you know without trying to guess.

Feedback

Feedback helps teams to identify areas for improvement and optimization their practices.

Courage

It can be interpreted as the preference for actions that aren't harmful to the team. Courage will help to reduce to optimize all organizational issues.

Respect

All members should respect each other and provide feedback that honors your relationship.

Advantages

- The main advantage of Extreme Programming is that this methodology allows software development companies to save costs and time required for project realization. Time savings are available because of the fact that XP focuses on the timely delivery of final products. Extreme Programming teams save lots of money because they don’t use too much documentation. They usually solve problems through discussions inside of the team.
- Simplicity is one more advantage of Extreme Programming projects. The developers who prefer to use this methodology create extremely simple code that can be improved at any moment.
- The whole process in XP is visible and accountable. Developers commit what they will accomplish and show progress.
- Constant feedback is also the strong side.It is necessary to listen and make any changes needed in time.

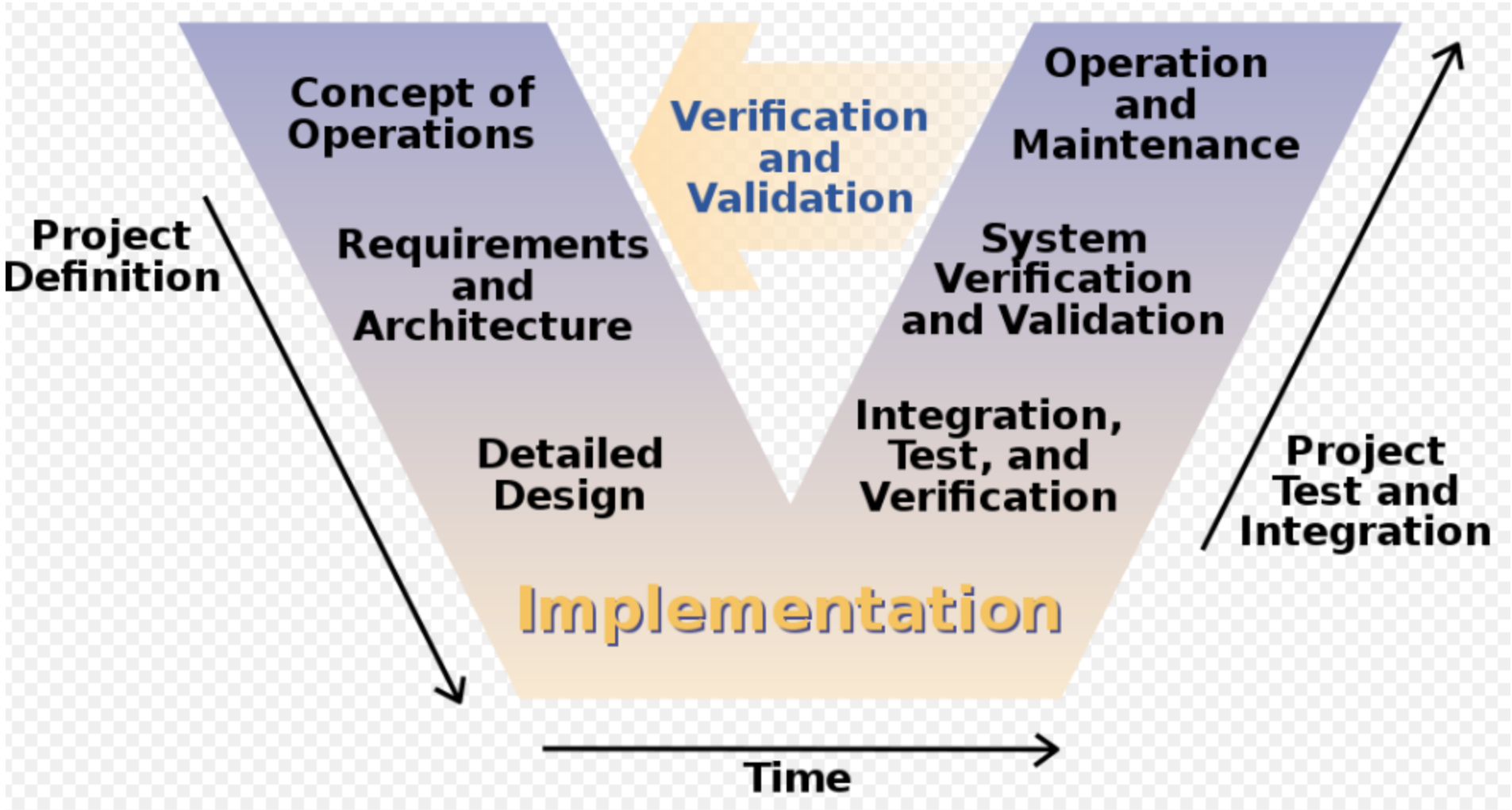
Disadvantages

- Some specialists say that Extreme Programming is focused on the code rather than on design. That may be a problem because good design is extremely important for software applications. It helps sell them in the software market. Additionally, in XP projects the defect documentation is not always good. Lack of defect documentation may lead to the occurrence of similar bugs in the future.

- One more disadvantage of XP is that this methodology does not measure code quality assurance. It may cause defects in the initial code.
- XP is not the best option if programmers are separated geographically.

V Shaped Model

- Also known as Verification and Validation model
- Extension of Waterfall Model
- Testing is associated with every phase of Lifecycle
- Verification Phase(Requirement analysis, System Design,Architecture design,Module design)
- Validation Phase(Unit testing,Integration,System,Acceptance,Testing)


Concept of Operations
Verification and Validation
Operation and Maintenance
Project Definition
Requirements and Architecture
System Verification and Validation
Detailed Design
Integration, Test, and Verification
Project Test and Integration
Implementation
Time

Advantages

- Time Saving
- Good understanding of project in the beginning.
- Every component must be testable
- Progress can be tracked easily.
- Proactive defect tracking

Disadvantages

- No feedback so less scope of changes
- Risk analysis is not done
- Not good for big or object-oriented projects.

Specialized Process Models

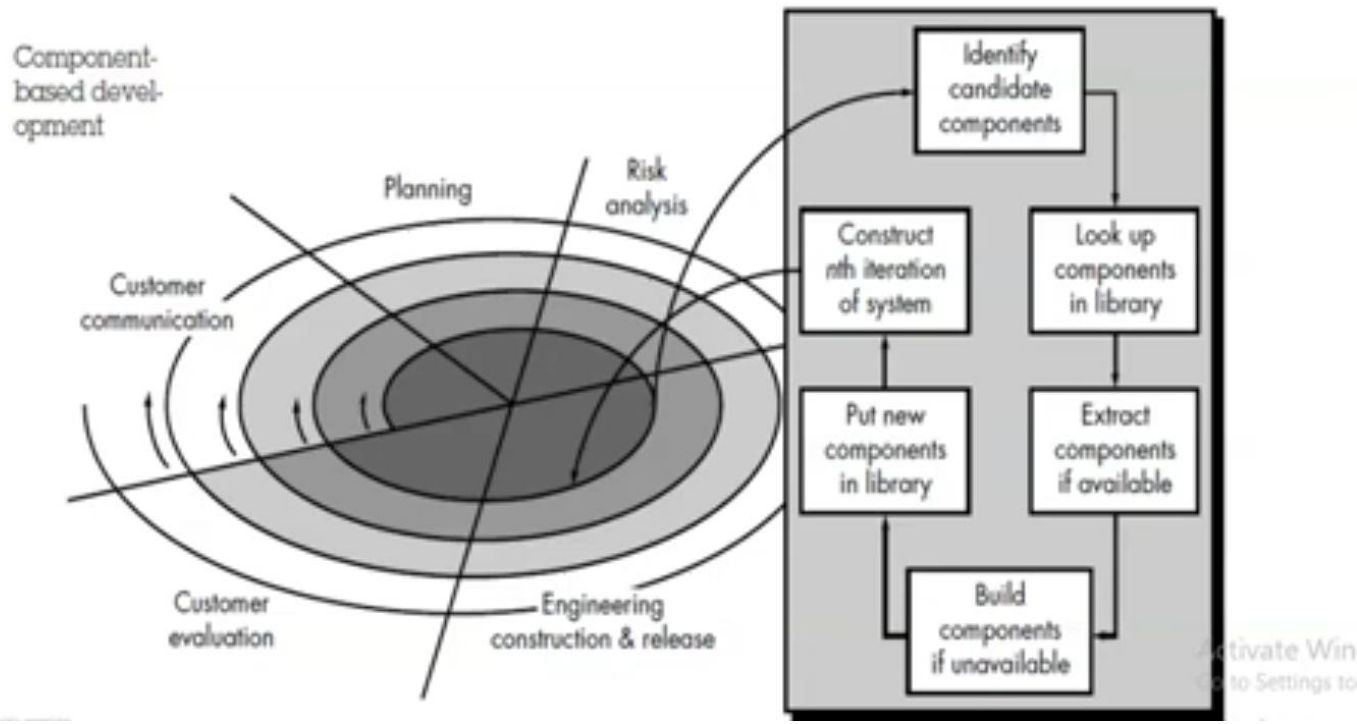
Specialized Process Models

Special process models take many features from one or more conventional models. However these special models tend to be applied when a narrowly defined software engineering approach is chosen.

Types in Specialized process models:

1. Component based development (Promotes reusable components)
2. The formal methods model (Mathematical formal methods are backbone here)
3. Aspect oriented software development (Uses crosscutting technology)

Component Base Development



The Formal Methods Models

The formal methods model encompasses a set of activities that leads to formal mathematical specification of software. Formal methods enable a software engineer to specify, develop and verify a computer system by applying rigorous mathematical notation.

When mathematical methods are used during software development, they provide a mechanism for eliminating many of the problems that are difficult to overcome using other software engineering paradigms.

Formal methods provide a basis for software verification and therefore enable software engineer to discover and correct errors that might otherwise go undetected.

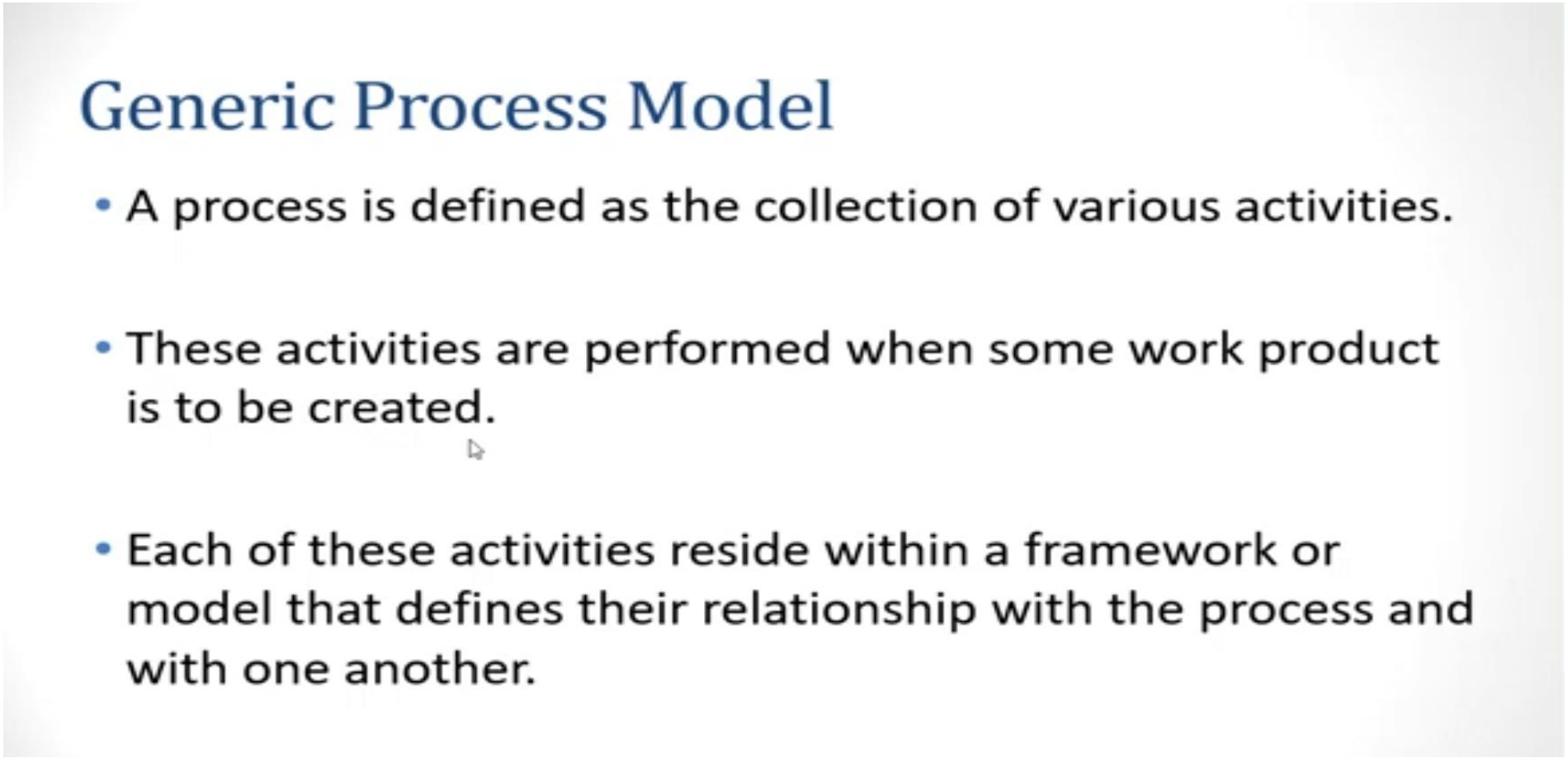
Aspect oriented Software Development

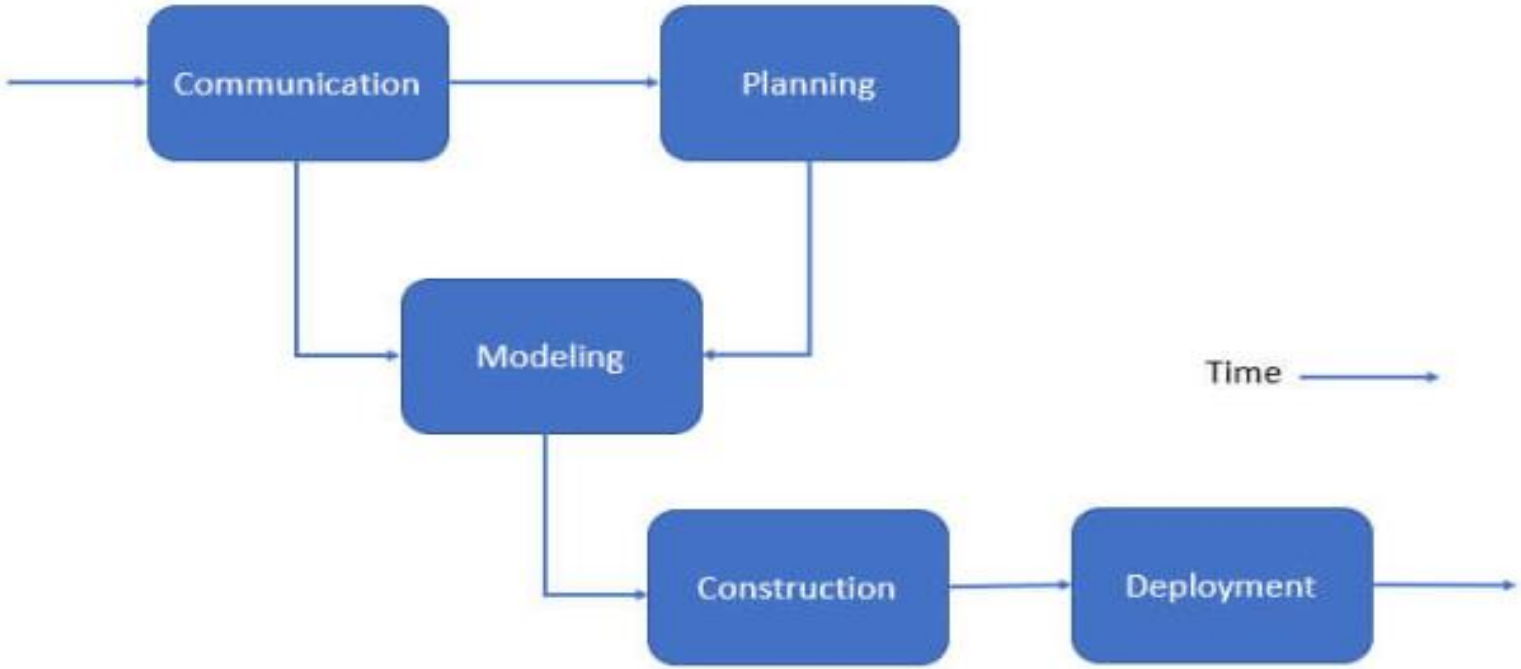
Aspect Oriented Software Development (AOSD) often referred to as aspect-oriented programming (AOP), a relatively new paradigm that provides process and methodology for defining, specifying designing and constructing aspects.

It addresses limitations inherent in other approaches, including object-oriented programming. AOSD aims to address crosscutting concerns by providing means for systematic identification, separation, representation and composition.

This results in better support for modularization hence reducing development, maintenance and evolution costs.

Generic Process Model





A software process is a collection of various activities.

There are five generic process framework activities:

1. Communication:
The software development starts with the communication between customer and developer.

2. Planning:
It consists of complete estimation, scheduling for project development and tracking.

3. Modeling:

- Modeling consists of complete requirement analysis and the design of the project like algorithm, flowchart etc.
- The algorithm is the step-by-step solution of the problem and the flow chart shows a complete flow diagram of a program.

4. Construction:

- Construction consists of code generation and the testing part.
- Coding part implements the design details using an appropriate programming language.
- Testing is to check whether the flow of coding is correct or not.
- Testing also check that the program provides desired output.

5. Deployment:

- Deployment step consists of delivering the product to the customer and take feedback from them.
- If the customer wants some corrections or demands for the additional capabilities, then the change is required for improvement in the quality of the software.

Generic Process Model

Umbrella Activities:

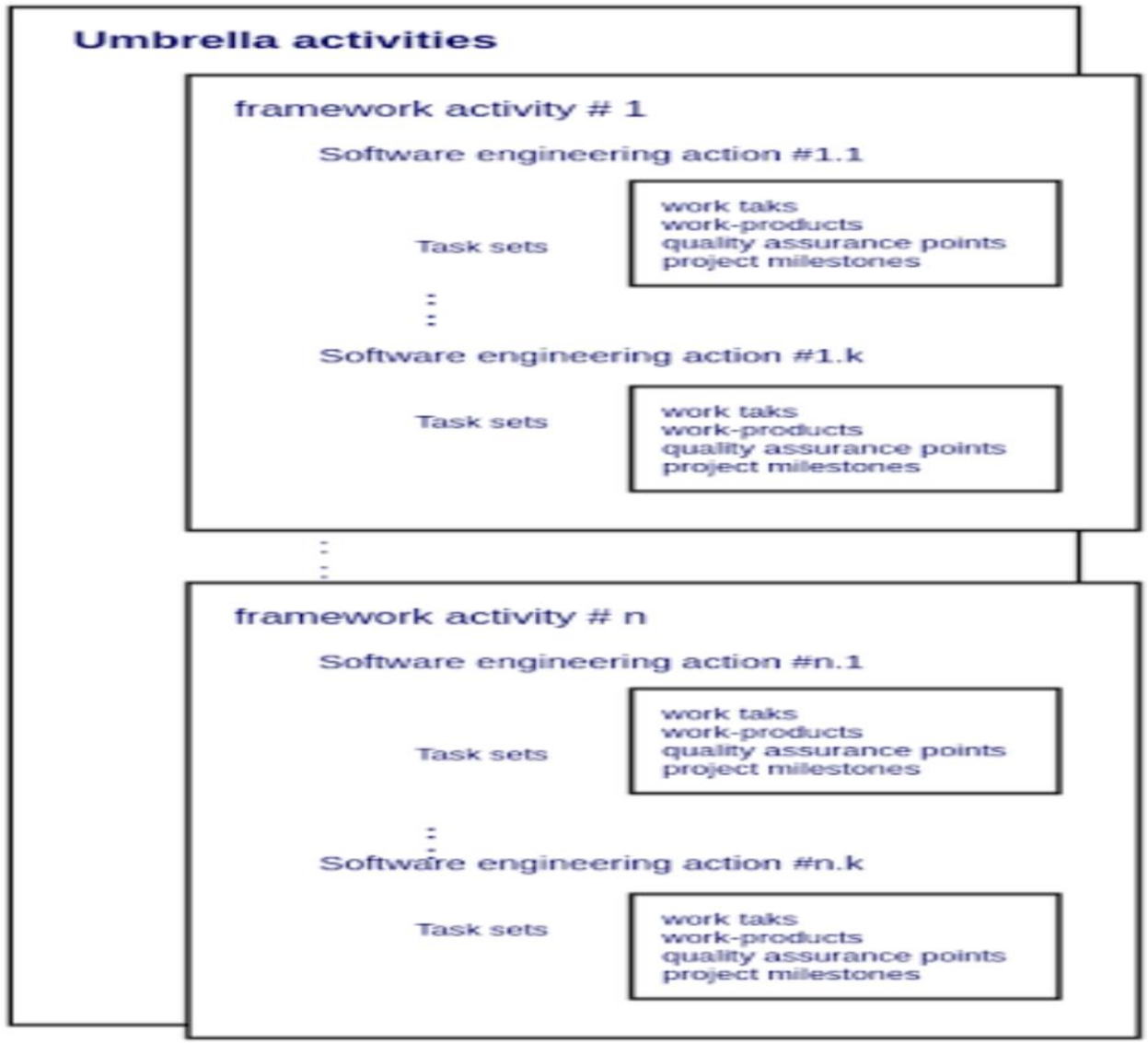
A set of umbrella activities—project tracking and control, risk management, quality assurance, and others—are applied throughout the process.

Process flow—describes how the framework activities and the actions and tasks that occur within each framework activity are organized with respect to sequence and time.

Generic Process Model

- A *linear process flow* executes each of the five framework activities in sequence.
- An *iterative process flow* repeats one or more of the activities before proceeding to the next.
- An *evolutionary process flow* executes the activities in a "circular" manner. Each circuit through the five activities leads to a more complete version of the software.
- A *parallel process flow* executes one or more activities in parallel with other activities

Process framework



Generic Process Model

- For a small software project requested by one person (at a remote location) with simple, straightforward requirements, the communication activity might encompass little more than a phone call with the appropriate stakeholder. Therefore, the only necessary action is *phone conversation,* and the work tasks (the *task set*) that this action encompasses are:

1. Make contact with stakeholder via telephone.
2. Discuss requirements and take notes.
3. Organize notes into a brief written statement of requirements.
4. E-mail to stakeholder for review and approval.

Generic Process Model

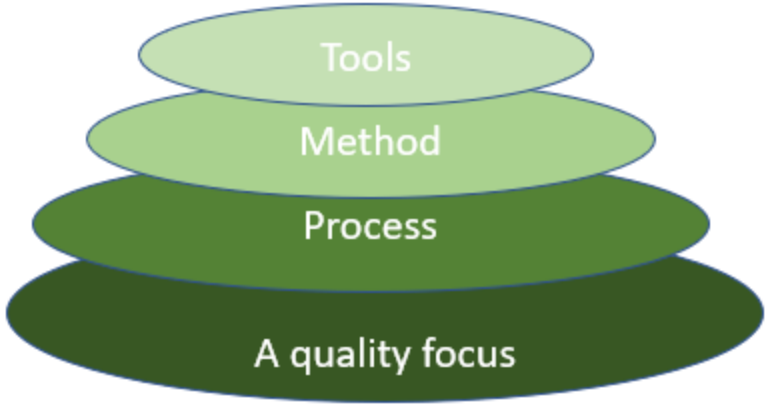
Task Set: software engineering action can be represented by several different *task sets*.

For a small, relatively simple project, the task set for requirements gathering might look like this:

1. Make a list of stakeholders for the project.
2. Invite all stakeholders to an informal meeting.
3. Ask each stakeholder to make a list of features and functions required.
4. Discuss requirements and build a final list.
5. Prioritize requirements.
6. Note areas of uncertainty.

Software Engineering as layered approach

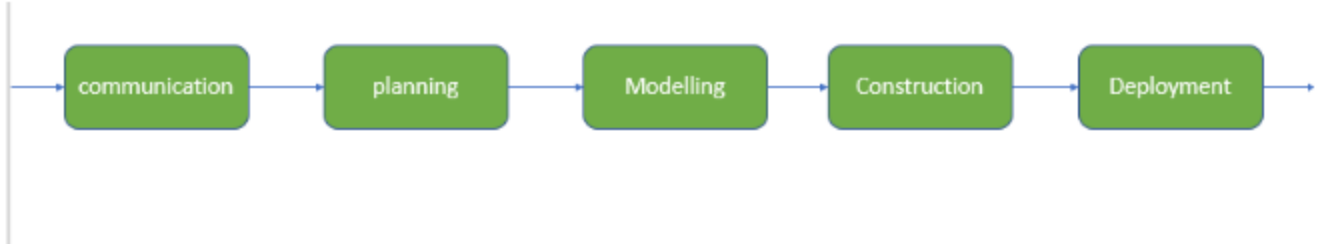
- Software Engineering is totally a layered technology.
- It means that to develop software one will have to go from one layer to another layer.
- This layered technology is consist of four layered which basically interact with each other as bottom to top.



Layered technology is divided into four parts:

1. A quality focus: It defines the continuous process improvement principles of software. It provides integrity that means providing security to the software so that data can be accessed by only an authorized person, no outsider can access the data. It also focuses on maintainability and usability.

2. Process: It is the foundation or base layer of software engineering. It is key that binds all the layers together which enables the development of software before the deadline or on time. Process defines a framework that must be established for the effective delivery of software engineering technology. The software process covers all the activities, actions, and tasks required to be carried out for software development.



Process activities are listed below:-

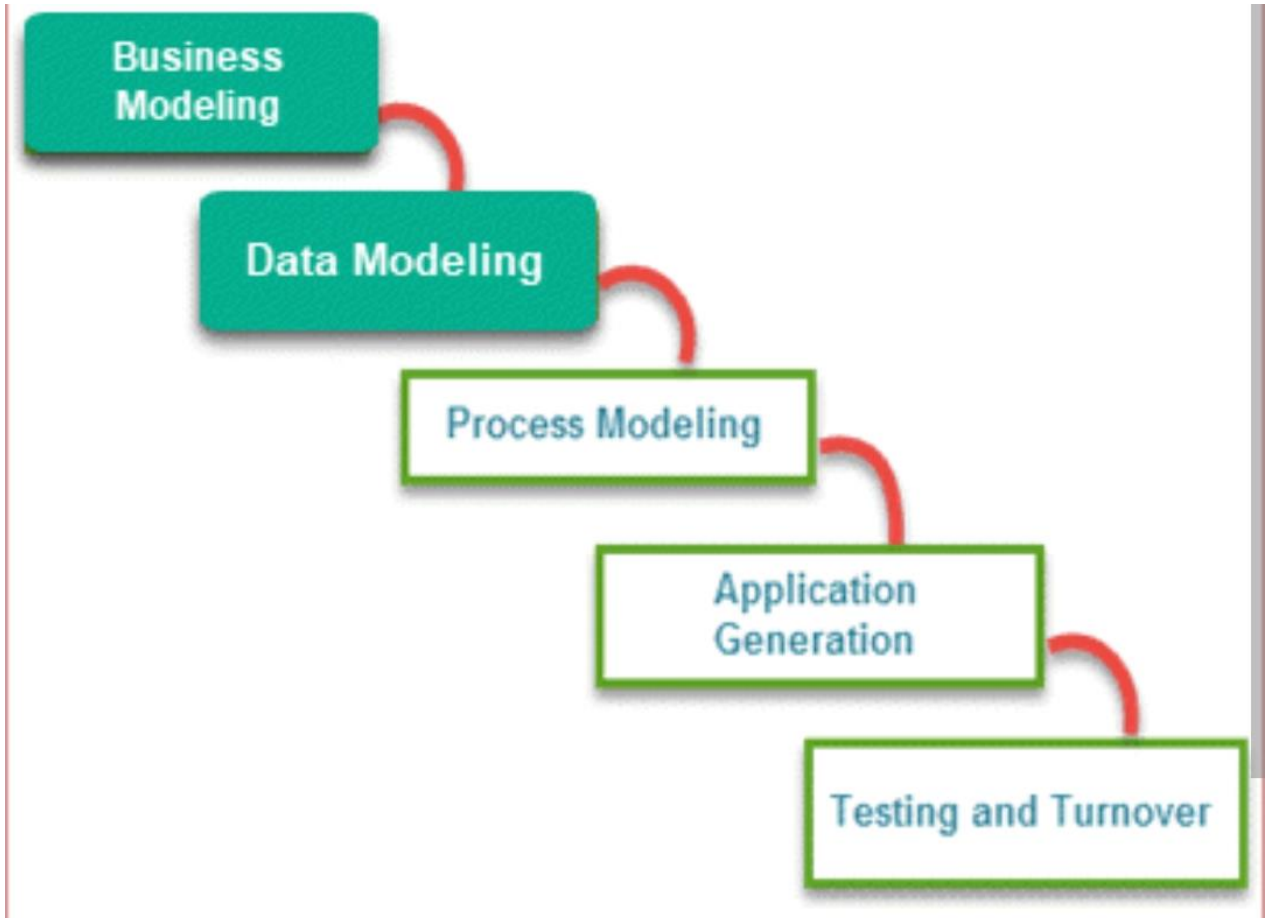
- **Communication:** It is the first and foremost thing for the development of software. Communication is necessary to know the actual demand of the client.
- **Planning:** It basically means drawing a map for reduced the complication of development.
- **Modeling:** In this process, a model is created according to the client for better understanding.
- **Construction:** It includes the coding and testing of the problem.
- **Deployment:-** It includes the delivery of software to the client for evaluation and feedback.

3. Method: During the process of software development the answers to all "how-to-do" questions are given by method. It has the information of all the tasks which includes communication, requirement analysis, design modeling, program construction, testing, and support.

4. Tools: Software engineering tools provide a self-operating system for processes and methods. Tools are integrated which means information created by one tool can be used by another.

RAD Model

- It is an adoption of the waterfall model
- It targets at developing software in a short span of time.
- It is based on prototyping and iterative development with no specific planning involved.
- Project can be broken down into small modules
- Each module can be assigned independently to separate teams
- Those modules can finally be combined to form the final product
- Development of each module involves the various basic steps as in waterfall model


Business Modeling
Data Modeling
Process Modeling
Application Generation
Testing and Turnover

RAD Model Phases

1. Business Modeling

- The information flow among business functions is defined by answering questions
- A complete business analysis is performed to find the information for business,

2. Data Modeling

- The data collected from business modeling is refined into a set of data objects (entities)
- The attributes of all data sets is identified and defined
- The relation between these data objects are established and defined in detail in relevance to the business model.

3. Process Modeling

- The data object sets defined in the Data Modeling phase are converted to establish the business information flow
- The process model for any changes or enhancements to the data object sets is defined

4. Application Generation

- The actual system is built
- Coding is done by using automation tools to convert process and data models into actual prototypes

5. Testing & Turnover

- The prototypes are independently tested during every iteration
- Hence overall testing time is reduced in the RAD model
- It reduces risk of major issues
- Overall testing is performed for testing integrated modules and interfaces

RAD Model Applications

- When the system should need to create the project that modularizes in a short span time (2-3 months)
- When the requirements are well-known
- When the technical risk is limited

RAD Model Advantages

- This model is flexible for change
- In this model, changes are adoptable
- Each phase in RAD brings highest priority functionality to the customer
- It reduced development time
- It increases the reusability of features

RAD Model Disadvantages

- It required highly skilled designers
- For smaller projects, we cannot use the RAD model
- On the high technical risk, it's not suitable
- Required user involvement.