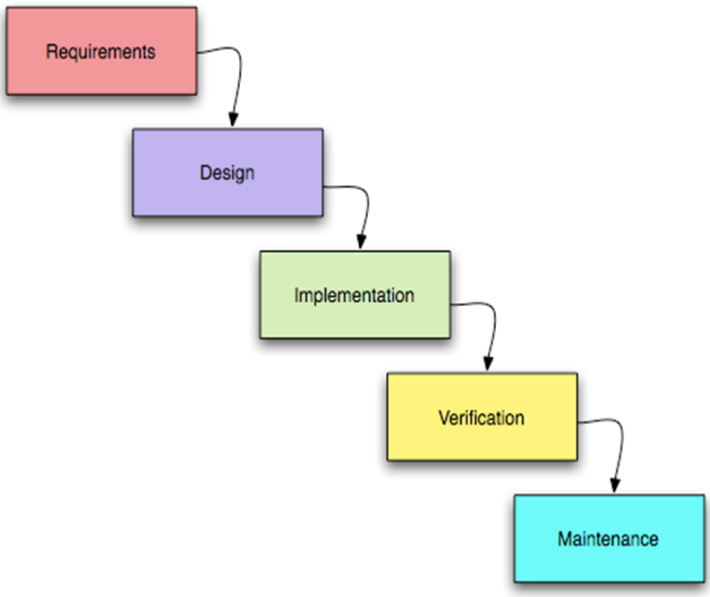
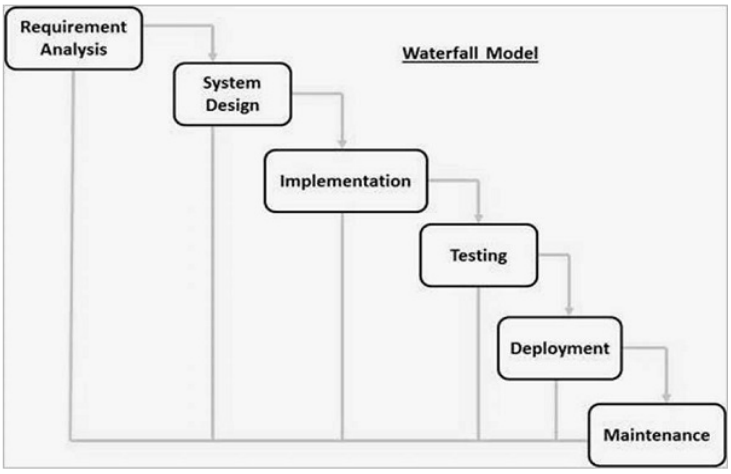
Waterfall development:

- The Waterfall model is the earliest SDLC approach that was used for software development.
- The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.
- The outcome of one phase acts as the input for the next phase sequentially.





Requirement Gathering and analysis − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

System Design − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

Implementation − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

Integration and Testing − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

Deployment of system − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

Maintenance − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

Waterfall Model - Application

Some situations where the use of Waterfall model is most appropriate are −

- Requirements are very well documented, clear and fixed.
- Technology is understood and is not dynamic.
- The project is short.

Waterfall Model - Advantages

The advantages of waterfall development are that

- It allows control. A schedule can be set with deadlines for each stage of development
- a product can proceed through the development process model phases one by one.
- Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance.
- Each phase of development proceeds in strict order.

Some of the major advantages of the Waterfall Model are as follows −

- Simple and easy to understand and use

- Easy to manage due to the rigidity of the model.
- Each phase has specific deliverables and a review process.

- Phases are processed and completed one at a time.

- Works well for smaller projects where requirements are very well understood.

- Clearly defined stages.

- Well understood milestones.

- Easy to arrange tasks.

- Process and results are well documented.

Waterfall Model - Disadvantages

- The disadvantage of waterfall development is that it does not allow much reflection or revision.
- Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The major disadvantages of the Waterfall Model are as follows −

- No working software is produced until late during the life cycle.

- High amounts of risk and uncertainty.

- Not a good model for complex and object-oriented projects.

- Poor model for long and ongoing projects.

- Not suitable for the projects where there is a high risk of changing.
- It is difficult to measure progress within stages.

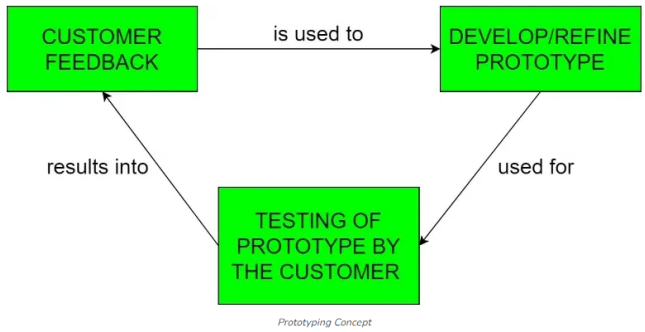
- Cannot accommodate changing requirements.

- Adjusting scope during the life cycle can end a project.

Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

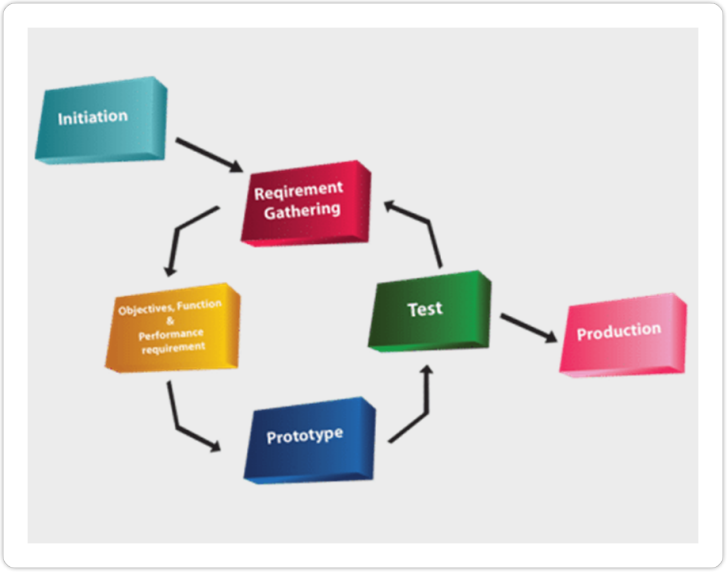
Prototyping Development:

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

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.



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 Test: 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: by following step 1—2,3,4 again.

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.

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 focus on prototype development may shift the focus away from the final product, leading to delays in the development process.
- 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.

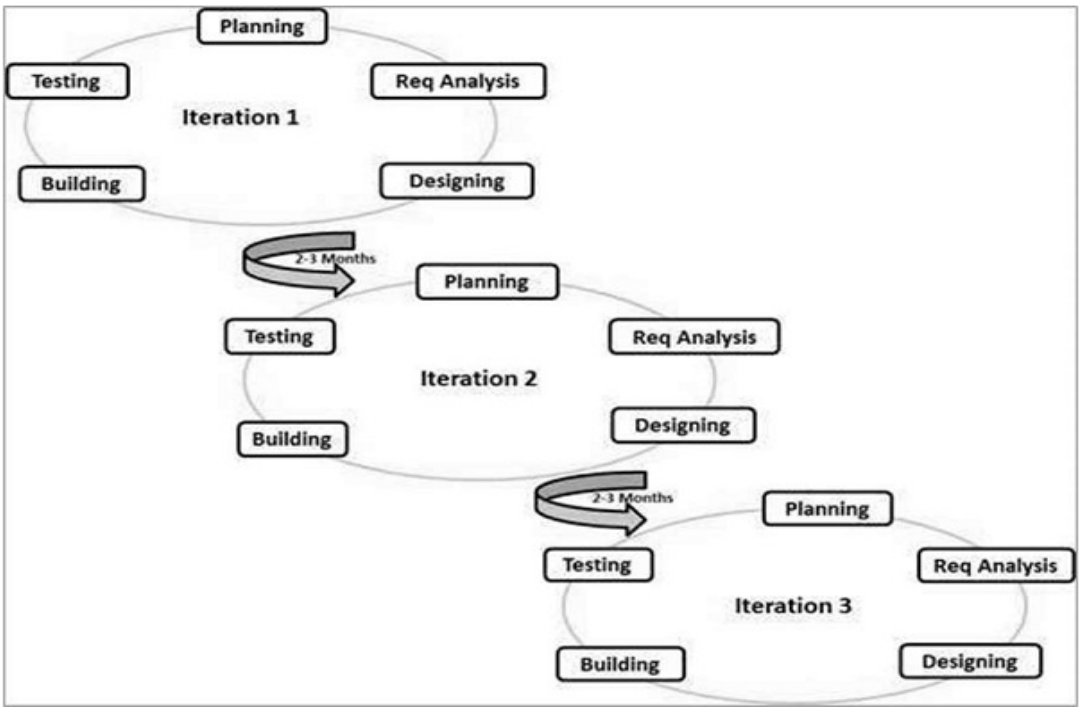
  For more software engineering models, you can refer to Classical Waterfall Model, Spiral Model, and Iterative Waterfall Model.

Agile development: https://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm

Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software products.

- break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks.
- Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.

LOOP: Plan- req analyze, design, code, deploy- plan



Following are the Agile Manifesto principles –

Individuals and interactions – In Agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.

Working software – Demo working software is considered the best means of communication with the customers to understand their requirements, instead of just depending on documentation.

Customer collaboration – As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.

Responding to change – Agile Development is focused on quick responses to change and continuous development.

Agile Model - Pros and Cons

The advantages of the Agile Model are as follows –

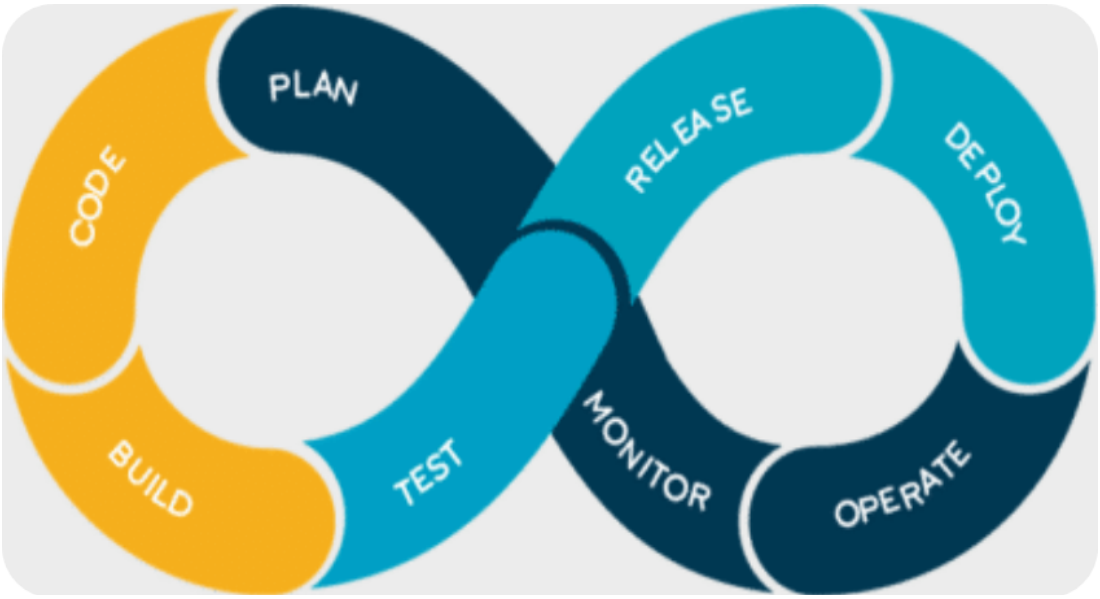
- Is a very realistic approach to software development.
- Promotes teamwork and cross training.
- Suitable for fixed or changing requirements
- Delivers early partial working solutions.
- Good model for environments that change steadily.
- Minimal rules, documentation easily employed.
- Little or no planning required.

- Easy to manage.
- Gives flexibility to developers.

The disadvantages of the Agile Model are as follows –

- Not suitable for handling complex dependencies.
- More risk of sustainability, maintainability
- Depends heavily on customer interaction, so if the customer is not clear team can be driven in the wrong direction.
- There is a very high individual dependency, since there is minimum documentation generated.
- Transfer of technology to new team members may be quite challenging due to lack of documentation.

DEV OPS:



1. Code: In this step, developers write the code for their application using any programming language or platform. For example, a developer working on a web application may write code in JavaScript, HTML, and CSS to create the frontend user interface.

Example: A developer writes code to create a login form for a website. They use HTML to structure the form, CSS to style it, and JavaScript to handle form validation and user interactions.

2. Build: After writing the code, developers build the application by compiling or packaging the code into an executable or deployable form. The specific build process depends on the programming language and tools used.

Example: In a Java application, developers use build tools like Apache Maven or Gradle to compile the source code into bytecode, package dependencies, and create a deployable JAR or WAR file.

3. Test: Testing is a crucial step in the DevOps life cycle. It involves running automated tests to validate the functionality, performance, and reliability of the application. Tools like Selenium, JUnit, or TestNG can be used for test automation.

Example: Automated tests are created to check if the login form mentioned earlier correctly handles valid and invalid inputs, verifies user authentication, and ensures the form elements are functioning as expected.

4. Release: Once the application has been thoroughly tested, it moves to the release phase. This involves managing, planning, scheduling, and controlling the deployment of the application to different environments such as development, staging, and production.

Example: The release process includes creating deployment packages, documenting release notes, and coordinating with the operations team to schedule the deployment of the tested application to a production environment.

5. Deploy: In this phase, the application artifacts or code files are deployed and executed on the target server or infrastructure. This step ensures that the application is available for use by end-users.

Example: The deployment process involves transferring the compiled code or deployment package to a web server, setting up the necessary configurations, and starting the application server to make the application accessible to users via a web URL.

6. Operate: After deployment, the application is up and running, and clients or end-users can start using it in real-world scenarios. This phase involves monitoring the application's performance, addressing user feedback, and handling any operational issues that may arise.

Example: Once the login form is deployed, users can access the website, enter their credentials, and log in. The operations team monitors the application's performance, response times, and server health to ensure smooth operation.

7. Monitor: This phase involves gathering crucial information about the application's performance and usage to ensure service uptime and optimal performance. Monitoring tools collect metrics and logs, allowing teams to identify and resolve any issues proactively.

Example: Monitoring tools track metrics such as server CPU usage, memory usage, response times, and error rates. They generate alerts if certain thresholds are exceeded or if anomalies are detected, enabling the team to investigate and resolve issues promptly.

8. Plan: The planning stage gathers information from the monitoring phase and uses it to implement changes and improvements for better performance. The feedback and insights gained through monitoring help in making informed decisions to enhance the application.