What is DevOps ?

Definition

- **DevOps** is a collection of two words, “**Development**” and “**Operations**,” representing a cultural approach that emphasizes collaboration and communication between development and operations teams to streamline the entire software delivery lifecycle.

- DevOps is the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes.

Dev

- Developers write code to create new features, programs, websites, or databases. They focus on producing new systems and applications as quickly as possible.

Ops

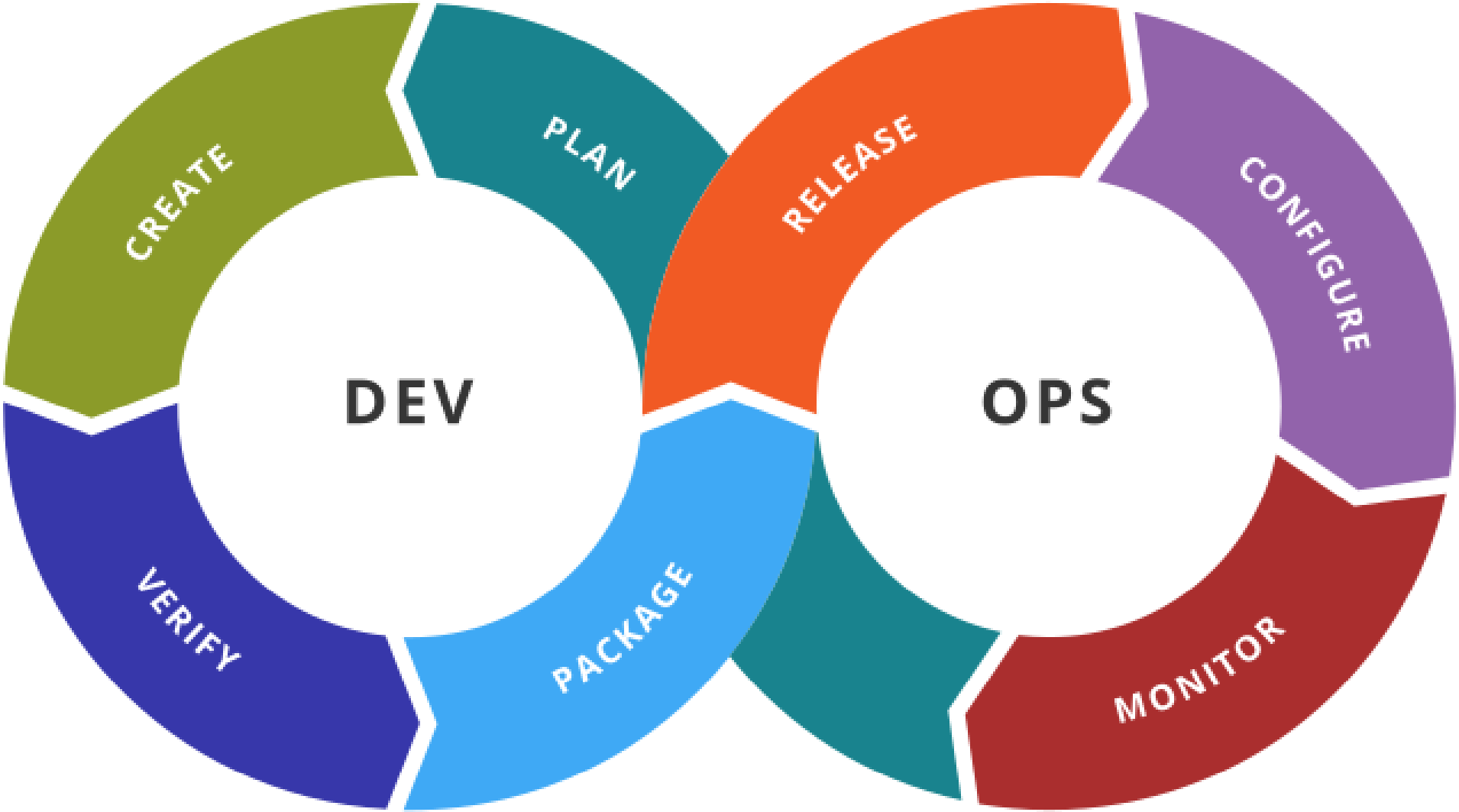
- Operators maintain infrastructure and ensure software runs smoothly. They deploy new releases into production and monitor live environments for issues. They focus on ensuring users have access to a fast, stable, and bug-free system.

How DevOps Works?

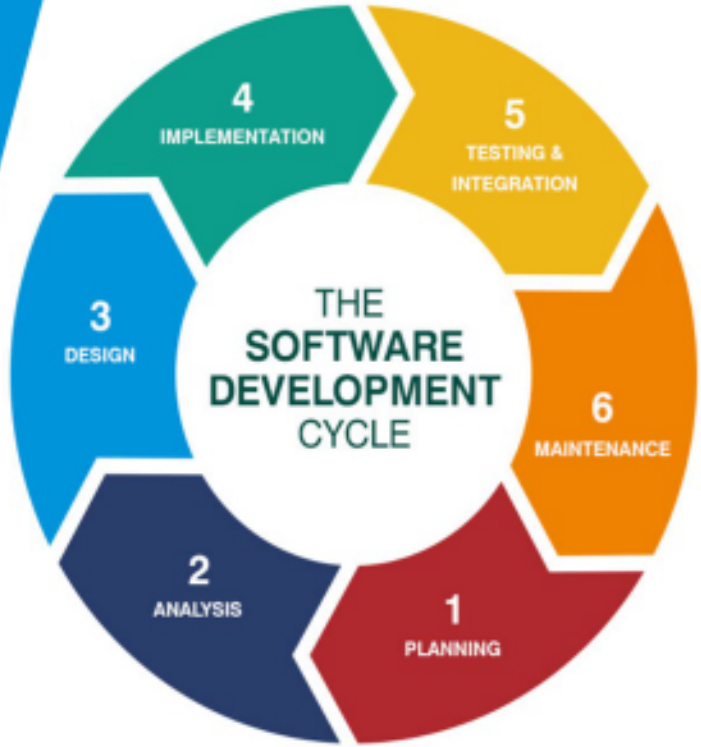
- Under a DevOps model, development and operations teams are no longer “siloed.” Sometimes, these two teams are merged into a single team where the engineers work across the entire application lifecycle, from development and test to deployment to operations, and develop a range of skills not limited to a single function.
- In some DevOps models, quality assurance and security teams may also become more tightly integrated with development and operations and throughout the application lifecycle. When security is the focus of everyone on a DevOps team, this is sometimes referred to as DevSecOps.

How DevOps Works?

- These teams use practices to automate processes that historically have been manual and slow. They use a technology stack and tooling which help them operate and evolve applications quickly and reliably.


CREATE
PLAN
RELEASE
CONFIGURE
DEV
OPS
VERIFY
PACKAGE
MONITOR

What is the Software Development Life Cycle?



SDLC (Software Development Life Cycle)

- A framework that describes the activities performed at each stage of a software development project.
- A systematic approach that generates a structure for the developer to design, create and deliver high-quality software based on customer requirements and needs. The primary goal of the SDLC process is to produce cost-efficient and high-quality products. The process comprises a detailed plan that describes how to develop, maintain, and replace the software.

Phases of SDLC

- Planning
- Analysis
- Design
- Implementation (Coding)
- Testing & Deployment
- Maintenance

Planning

- The first stage of SDLC is all about “What do we want?” Project planning is a vital role in the software delivery lifecycle since this is the part where the team estimates the cost and defines the requirements of the new software.

- Planning, involves defining the software's purpose and scope. The team collaborates to understand the end-users' needs and the goals the software should meet. Essentially, we ask, "What problem will this software solve?" and "What value will it offer to the user?"

Planning

- A feasibility study also takes place during the Planning phase. Developers and product teams evaluate technical and financial challenges that might affect the software's development or success.

- Key documents such as the Project Plan and Software Requirement Specification (SRS) are created.

- The Planning phase fosters effective communication and collaboration within the team. By defining clear roles, responsibilities, and expectations, it lays a solid foundation for an efficient software development process.

Analysis

- The second step of SDLC is gathering maximum information from the client requirements for the product. Discuss each detail and specification of the product with the customer.

- The development team will then analyze the requirements keeping the design and code of the software in mind. Further, investigating the validity and possibility of incorporating these requirements into the software system.

- The main goal of this stage is that everyone understands even the minute detail of the requirement. Hardware, operating systems, programming, and security are to name the few requirements.

Analysis

- The project team collects information from stakeholders, including analysts, users, and clients. They conduct interviews, surveys, and focus groups to understand the user's expectations and needs. The process involves not only asking the right questions but also accurately interpreting the responses.

- After collecting the data, the team analyzes it, distinguishing the essential features from the desirable ones. This analysis helps the team understand the software's functionality, performance, security, and interface needs.

- These efforts result in a Requirements Specification Document.

Design

- The Design phase is all about building the framework. Key activities include crafting data flow diagrams, constructing entity-relationship diagrams, and designing user interface mock-ups. The team also identifies system dependencies and integration points. They also set the software's limitations, such as hardware constraints, performance requirements, and other system-related factors.

- The culmination of these tasks is an exhaustive Software Design Document (SDD). This document serves as the roadmap for the team during the coding phase. It meticulously details the software's design, from system architecture to data design, and even user interface specifics.

Coding

- The Coding phase in the Software Development Life Cycle (SDLC) is when engineers and developers get down to business and start converting the software design into tangible code.
- This development phase aims to develop software that is functional, efficient, and user-friendly. Developers use an appropriate programming language, Java or otherwise, to write the code, guided by the SDD and coding guidelines. This document, acting as a roadmap, ensures the software aligns with the vision set in earlier phases.
- Another key aspect of this phase is regular code reviews. Team members carefully examine each other's work to identify any bugs or inconsistencies.

Testing

- Once the developers build the software, then it is deployed in the testing environment. Then the testing team tests the functionality of the entire system. In this fifth phase of SDLC, the testing is done to ensure that the entire application works according to the customer's requirements.
- After testing, the QA and testing team might find some bugs or defects and communicate the same with the developers. The development team then fixes the bugs and send it to QA for a re-test. This process goes on until the software is stable, bug-free and working according to the business requirements of that system.

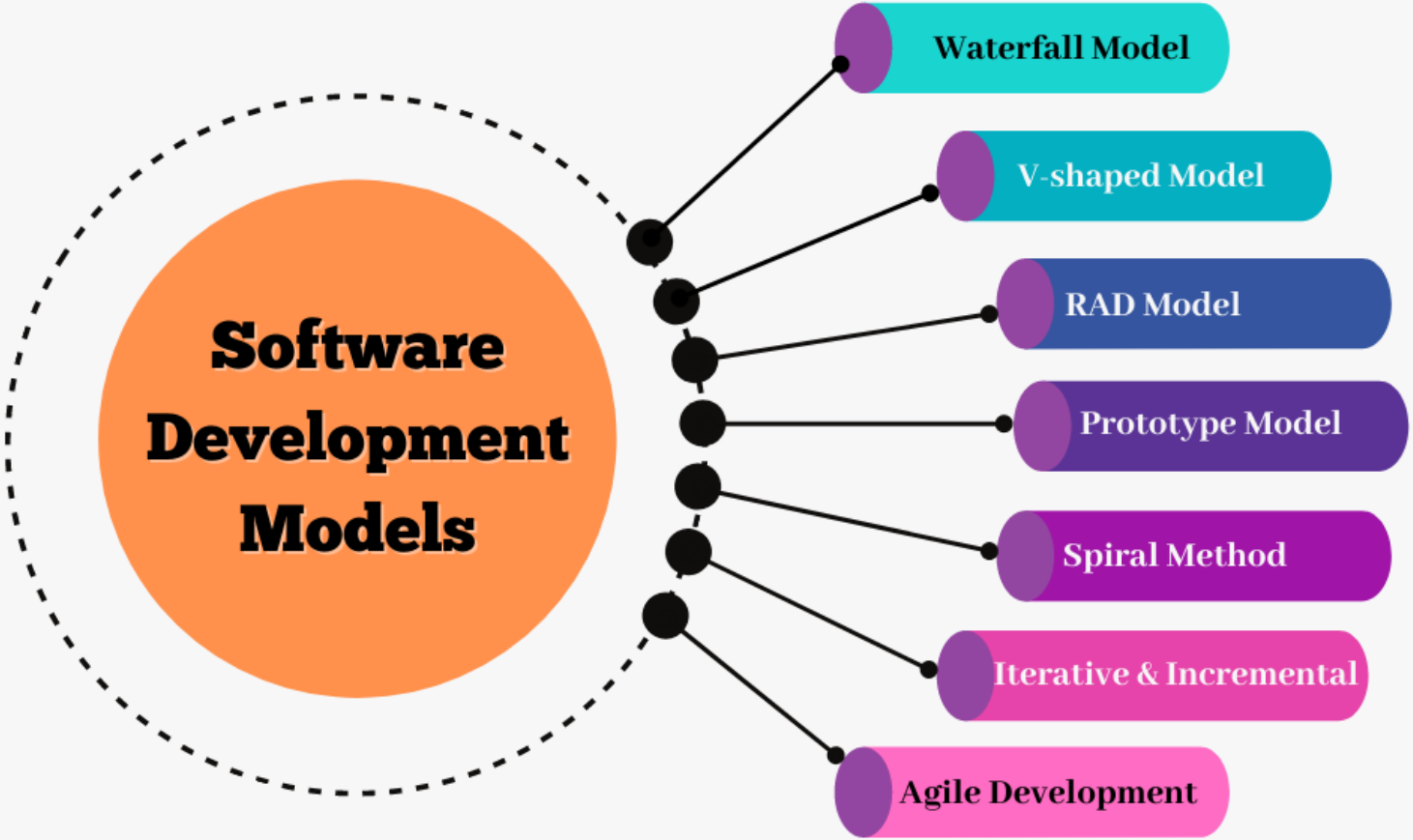
Deployment

- The sixth phase of SDLC: Once the testing is done, and the product is ready for deployment, it is released for customers to use. The size of the project determines the complexity of the deployment.
- The users are then provided with the training or documentation that will help them to operate the software. Again, a small round of testing is performed on production to ensure environmental issues or any impact of the new release.

Maintenance

- The maintenance phase is characterized by constant assistance and improvement, which guarantees the software's best possible functioning and longevity and ensures it meets customer expectations.
- The primary focus is to adapt to the software's changing needs. This adaptation involves responding to user feedback, resolving unexpected issues, and upgrading the software based on users' evolving requirements. It's a continuous process of refining and adapting.

AGILE INFOWAYS



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.

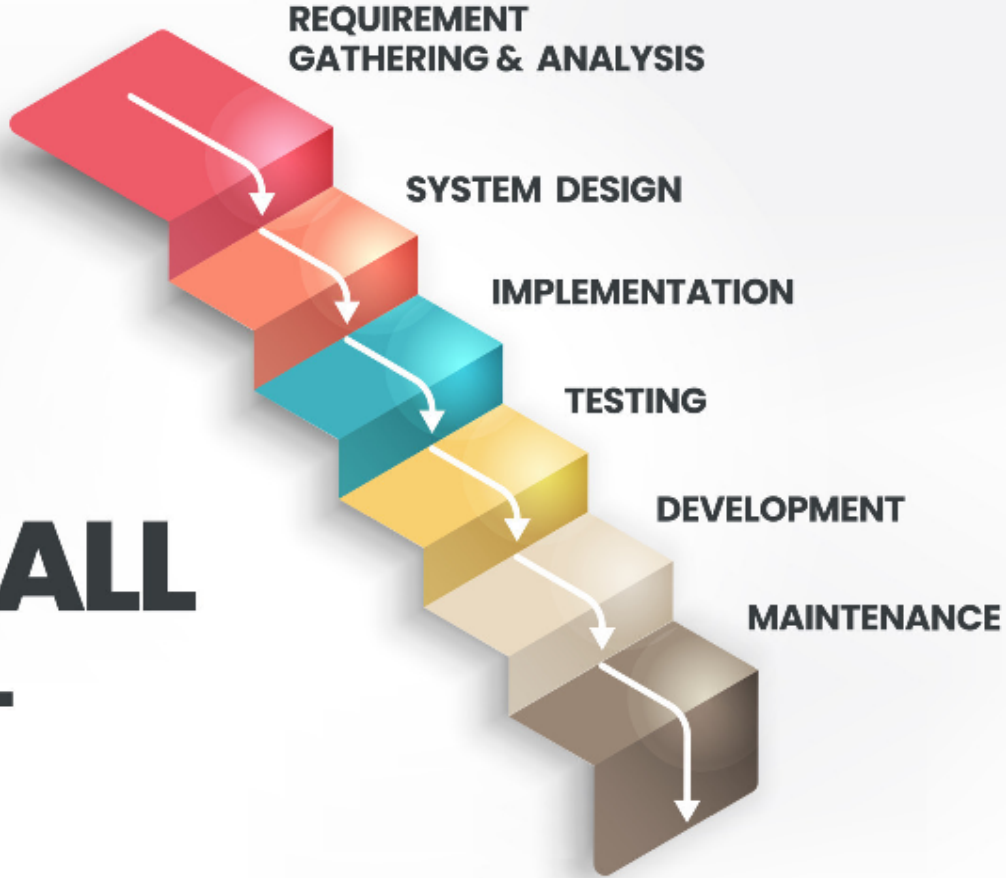
Advantages:

- The waterfall model is a simple model that 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, leading to no complexity and making the project easily manageable.

Disadvantages:

- The 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.
- The waterfall model cannot be used for projects that have uncertain requirements or wherein the requirement keeps on changing as 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.

WATERFALL MODEL



V model

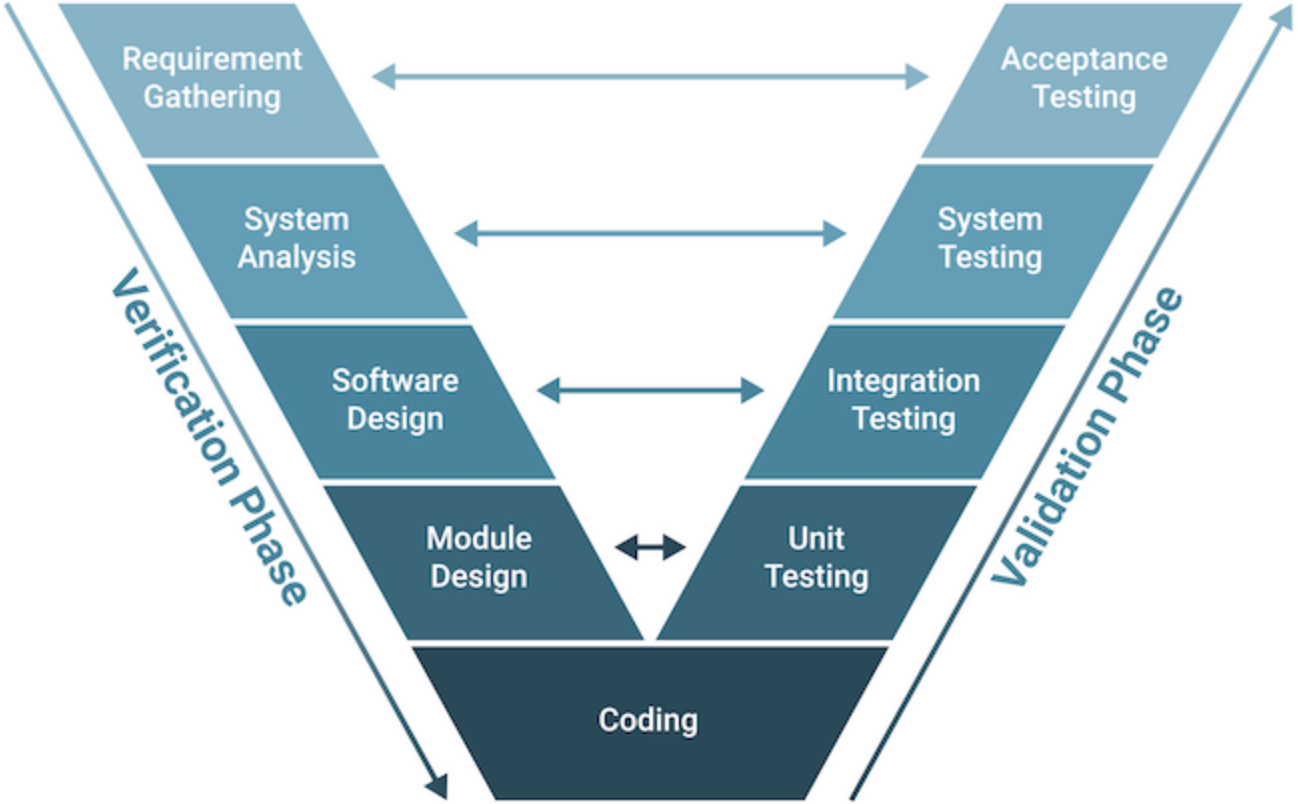
V- Model is also known as Verification and Validation Model. In this model Verification & Validation goes hand in hand i.e. development and testing goes parallel. V model and waterfall model are the same except that the test planning and testing start at an early stage in V-Model.

Advantages:

- 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:

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


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

Unit Testing: Focuses on testing individual components or units of the software in isolation.

Integration Testing: Tests the interactions between integrated modules or components.

System Testing: Validates the complete and integrated system against specified requirements.

Acceptance Testing: Verifies whether the software meets the business requirements and is ready for release.

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 compared to the actual software. Dummy functions are used to create prototypes. This is a valuable mechanism for understanding the customers' needs.

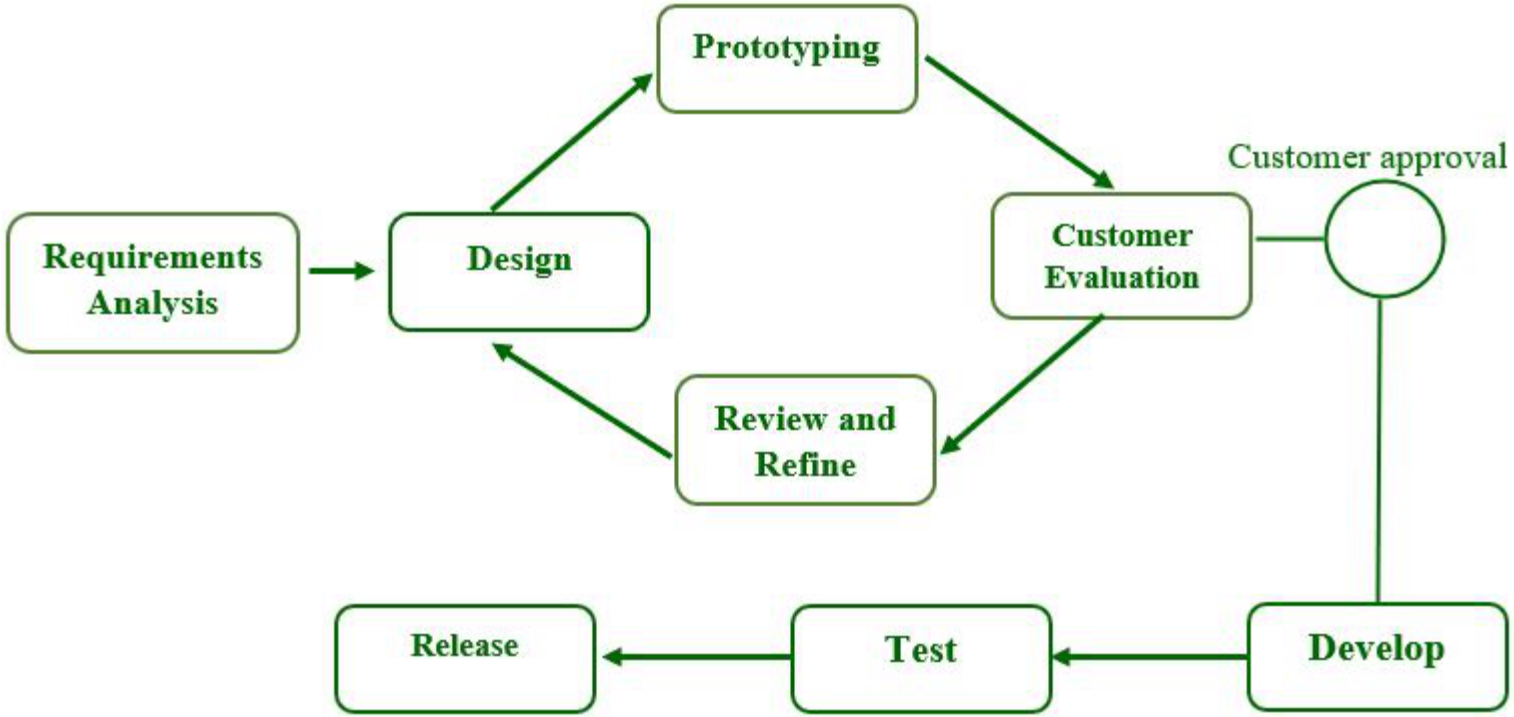
Software prototypes are built prior to the actual software to get valuable feedback from the customer. Feedbacks are implemented and the prototype is again reviewed by the customer for any change. This process goes on until the model is accepted by the customer.

Advantages:

- Prototype model reduces the cost and time of development as the defects are found much earlier.
- Missing features 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:

- 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.


Prototyping
Customer approval
Requirements Analysis
Design
Customer Evaluation
Review and Refine
Release
Test
Develop

Spiral model

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

Spiral model

- Planning

The first phase of the Spiral Model is the planning phase, where the scope of the project is determined and a plan is created for the next iteration of the spiral.

- Risk Analysis

In the risk analysis phase, the risks associated with the project are identified and evaluated. Risks can be operational (e.g., team changes), technical (e.g., untested technologies), or external (e.g., regulatory changes).

- Engineering

In the engineering phase, the software is developed based on the requirements gathered in the previous iteration.

- Evaluation

In the evaluation phase, the software is evaluated to determine if it meets the customer's requirements and if it is of high quality.

Contd..

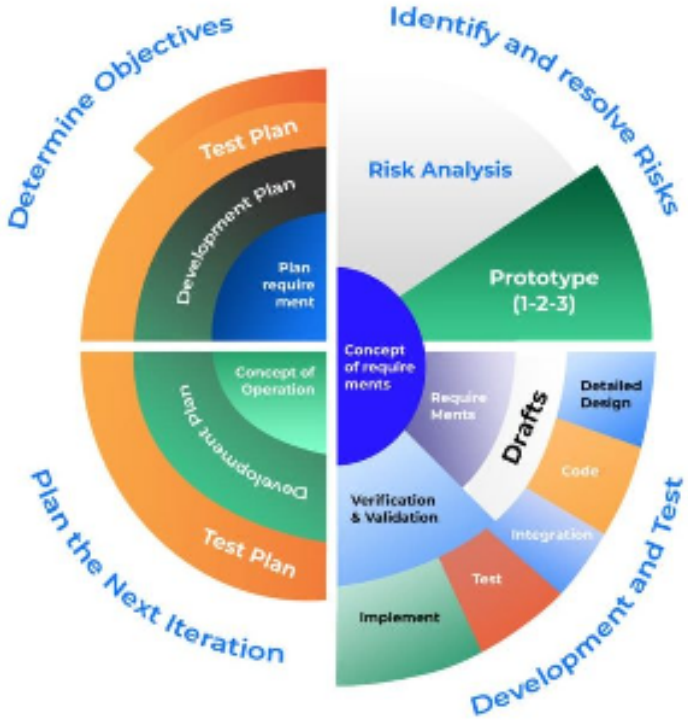
Advantages:

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

Disadvantages:

- 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.

Spiral model



Iterative & Incremental model

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.

Phases of Iterative & Incremental Development Model:

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

Iterative & Incremental model

Inception Phase

- Define the project scope and its feasibility.
- Establish a high-level understanding of the system to be built.

Elaboration Phase

- Refine the project vision and requirements.
- Establish a robust architecture for the system.

Construction Phase

- Build the software incrementally based on the established architecture.
- Develop a fully functional system.

Transition Phase

- Deploy the system into the production environment.
- Ensure the system meets the users' needs.

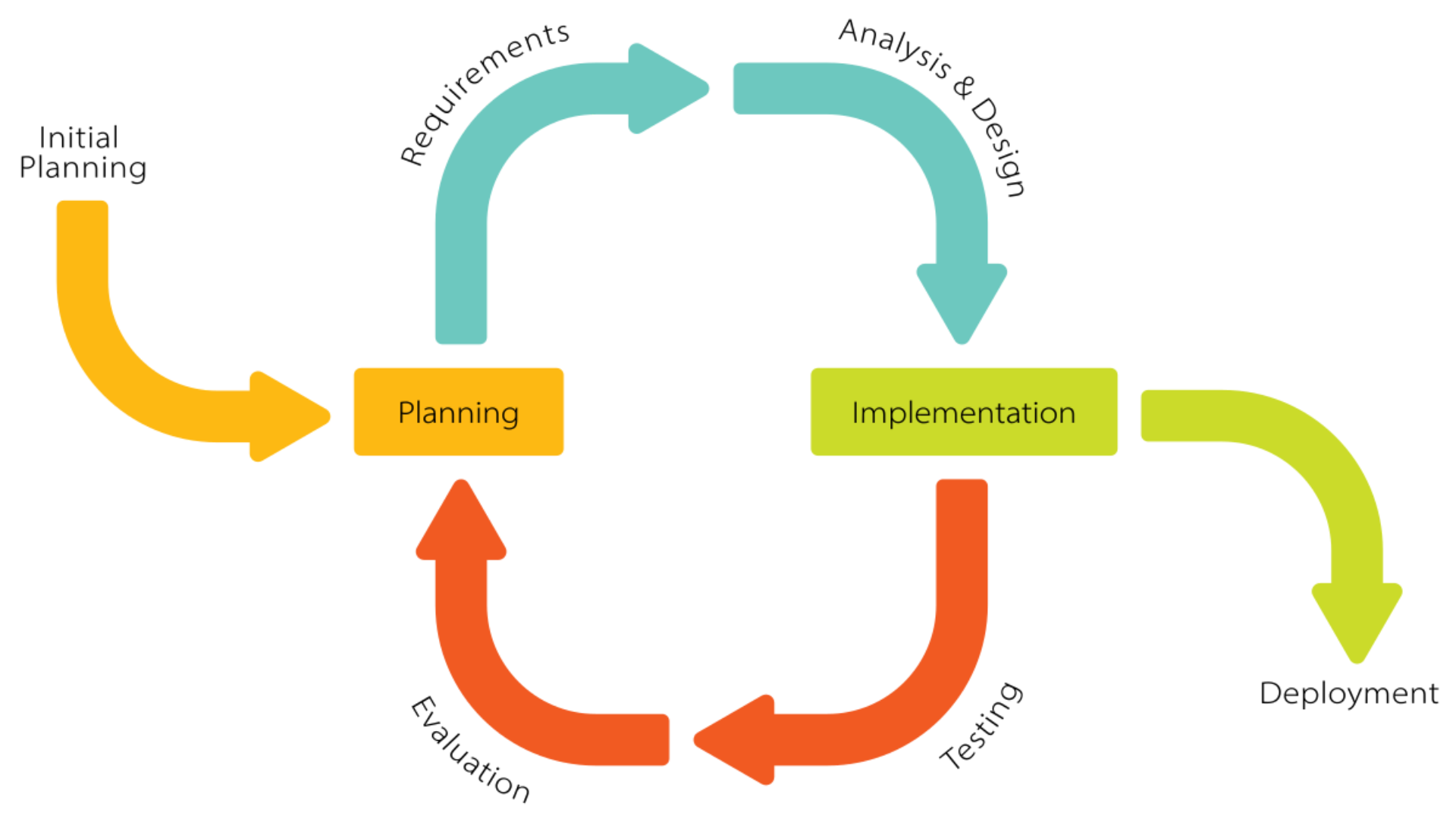
Contd..

Advantages:

- 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:

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


Initial Planning
Requirements
Analysis & Design
Planning
Implementation
Deployment
Testing
Evaluation

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.

Contd..

Advantages:

- 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:

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


Time
Efforts
Resources
BIG
BANG
Software

Agile model

- The meaning of Agile is swift or versatile. "Agile process model" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts that do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.
- Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

Agile model

- The agile software development life cycle is a software development project methodology that prioritizes adaptability, flexibility, rapid development, and transformation.
- 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 the previous functionality.
- In agile iterations are termed as sprints. Each sprint lasts for 2-4 weeks. At the end of each sprint, the product owner verifies the product and after his approval, it is delivered to the customer.

Phases of Agile model

- Requirements gathering
- Design the requirements
- Construction/ iteration
- Testing/ Quality assurance
- Deployment
- Feedback

Planning & Requirements gathering

- The planning phase in Agile is about defining the overall vision and high-level objectives of the project. The Product Owner works with stakeholders to create a Product Backlog, which is a prioritized list of features or tasks to be completed. This backlog is dynamic and evolves over time as new requirements emerge.

Design

- During this phase, the team focuses on designing the system architecture and features based on the requirements defined in the backlog. However, Agile design is adaptive and can change as the project progresses, unlike traditional SDLCs where the design is typically fixed early on.

Construction/ iteration

- This phase involves the actual coding and implementation of the features identified in the product backlog. Teams typically work in sprints (usually 1-4 weeks), and each sprint results in a potentially shippable product increment.

Testing

- Testing in Agile is continuous and happens throughout the development process, rather than as a separate phase at the end. This ensures that issues are caught early, and the product evolves in a stable, reliable manner.

Deployment

- Once the software is developed, tested, and reviewed, it is deployed to production. In Agile, deployment can happen frequently, even after every sprint, ensuring that the product is always ready for release.

Maintenance

- After the software is deployed, maintenance is an ongoing activity. Agile allows teams to adapt and respond to feedback from users and stakeholders, and they can address issues or add new features in subsequent iterations.

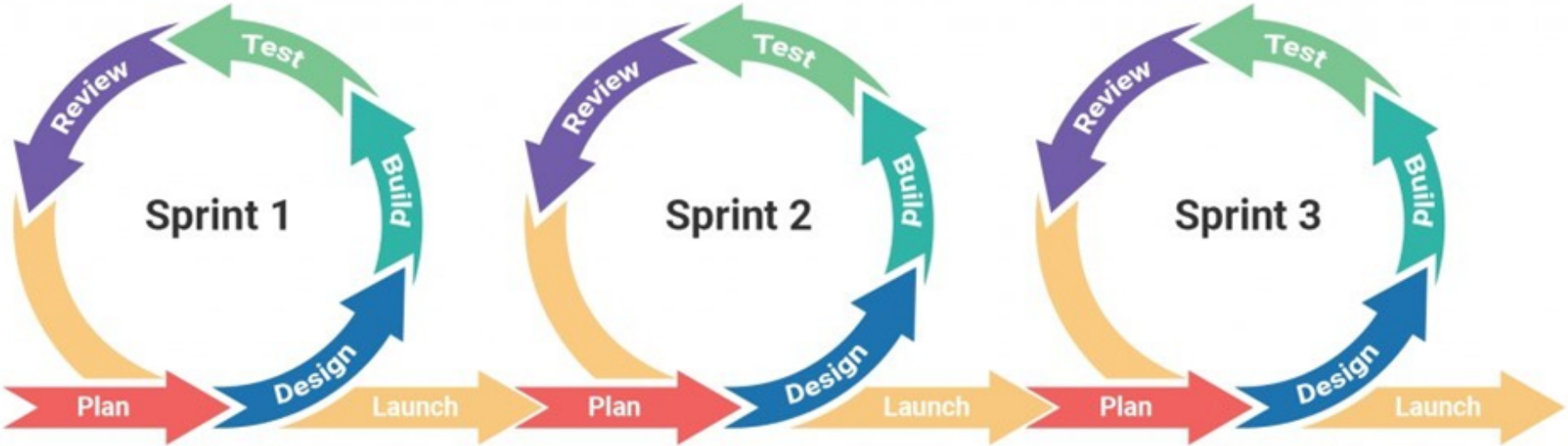
Contd..

Advantages of Agile Model:

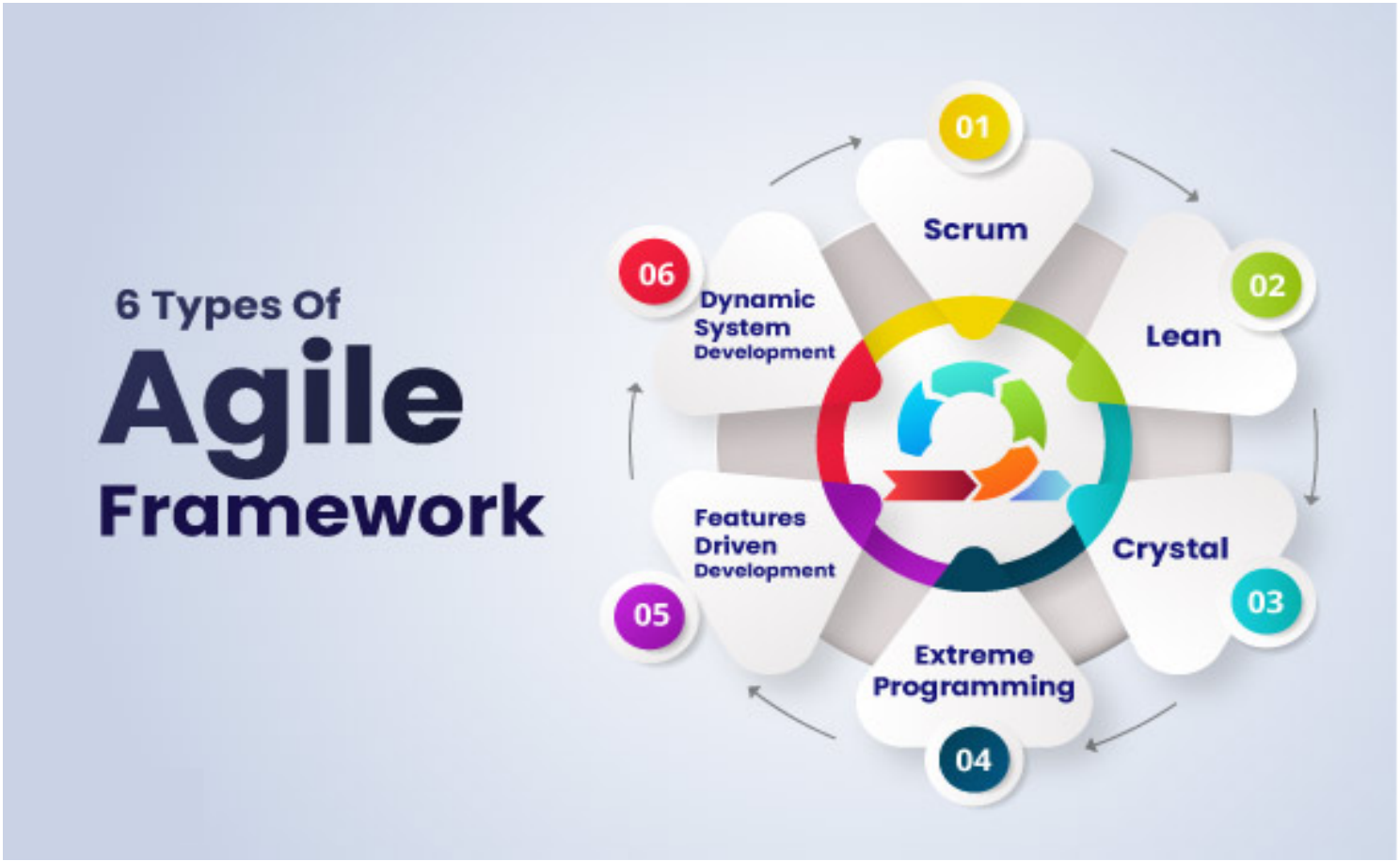
- It allows more flexibility to adapt to the changes.
- The new feature can be added easily.
- Customer satisfaction as the feedback and suggestions are taken at every stage.

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.


Review
Test
Build
Sprint 1
Design
Plan
Launch
Review
Test
Build
Sprint 2
Design
Plan
Launch
Review
Test
Build
Sprint 3
Design
Plan
Launch

Types of Agile Framework



- If you choose an agile software project model, the second step is to choose the framework based on your development team, workflow, scale, project complexity, requirements, targets, and needs.

Scrum

Scrum is used to do tasks in a scenario where teams are involved. The idea of scrum comes from the game of rugby and it helps in the proper working of smaller teams. The three aspects of scrum can be described as below:

- Scrum Master helps in building the team, look after sprint meetings and take care of problems in the path of progress.
- Product Owner helps in building the product backlog. The owner also gives the most priority to the backlog and ensures that there is delivery after every iteration.
- The Scrum Team looks after their work and completes the sprint.

Scrum

Key Roles

- Product Owner: Manages the product backlog and prioritizes features.
- Scrum Master: Facilitates the Scrum process and removes impediments.
- Development Team: Cross-functional team that delivers the product increment.

Key Artifacts

- Product Backlog: List of prioritized work items (user stories, tasks).
- Sprint Backlog: Subset of the product backlog selected for a sprint.
- Increment: The working product delivered at the end of a sprint.

Scrum

Services:

- **Sprint Planning**: Plan what work will be completed during the sprint.
- **Daily Standup**: Short daily meeting to discuss progress, plans, and blockers.
- **Sprint Review**: Showcase the completed work to stakeholders.
- **Sprint Retrospective**: Reflect on the sprint to identify improvements.

Lean

Lean Software Development is an Agile framework that applies principles from **Lean manufacturing** to the software development process. Its primary focus is on delivering maximum value to the customer by eliminating waste, optimizing processes, and fostering continuous improvement.

Principles of Lean Software Development

Eliminate Waste - Identify and remove anything that does not add value to the customer.

Build Quality In - Prevent defects rather than fixing them later.

Create Knowledge - Emphasize learning and adapting through feedback and experimentation.

Defer Commitment - Make decisions as late as possible to allow for maximum flexibility and adaptability.

Lean

Deliver Fast - Minimize lead time and deliver value to customers as quickly as possible.

Respect People - Empower individuals and teams to contribute their best work.

Optimize the Whole - Focus on the entire system rather than optimizing individual components.

Extreme Programming

This methodology is helpful when the demand is not stable i.e. the requirement of the demand is always changing. It can be also used when there is uncertainty regarding the functionality of the whole system. Thus, with the help of extreme programming, the product is released in shorter intervals of time thus creating checkpoints and improving productivity. The aim of extreme programming works with respect to the customer.

The different phases of extreme programming are:

- Planning
- Analysis
- Execution
- Wrapping
- Closure

Kanban method

- The Japanese word Kanban refers to a card that has all the information which is required and done on the product during its process of completion. This method is widely used in software development.
- Kanban is a visual framework for managing work that focuses on optimizing the flow of tasks through a system. It does not prescribe time-boxed iterations.
- **Visualize Workflow**: Use a Kanban board with columns (e.g., To Do, In Progress, Done).
- **Limit Work in Progress (WIP)**: Set limits to prevent bottlenecks and improve flow.
- **Manage Flow**: Continuously monitor and optimize the flow of tasks.
- **Continuous Improvement**: Use feedback and metrics to improve the process.

Feature-Driven Development (FDD)

- FDD focuses on delivering tangible, working software features in a structured way. It combines iterative development with modeling and design.

Key Practices:

- Develop an overall model of the system.
- Build a feature list (a breakdown of system functionality).
- Plan by feature and deliver in small increments.
- Design and build features iteratively.

Crystal

- Crystal is a family of Agile frameworks (e.g., Crystal Clear, Crystal Orange, Crystal Red) that adapt based on team size, system criticality, and project priorities.

Key Practices:

- Frequent delivery of working software.
- Close communication within the team.
- Reflective improvement (learning from experiences).
- Tailored processes based on project needs.

Dynamic Systems Development Method (DSDM)

- DSDM is an Agile framework focused on delivering projects on time and within budget while emphasizing active user involvement.

Key Practices:

- MoSCoW Prioritization: Categorize requirements into Must Have, Should Have, Could Have, and Won't Have.
- Timeboxing: Fixed deadlines for delivering specific functionality.
- Iterative development and prototyping.
- Continuous user involvement.

Agile Principles

- Satisfy the customer with recurring delivery of software at regular intervals.
- If there are changes in requirements then accept them, even if they arrive late in the development cycle
- Deliver the software at intervals between 3 weeks to 3 months.
- The developers and the businessmen should work in collaboration.
- Building projects keeping in mind individuals who are highly motivated and then supporting them to get the work done.
- Using face to face conversation as the most effective way of communication.
- To measure progress the software should be working.
- The development should be sustainable and there should be consistency.
- Technical excellence and a good design help in better agility.
- Simplicity is the key.
- A team which is self-organizing helps to deliver best architecture, design and requirements.
- The team decides and discusses ways in which the team can be more effective.

Contd..

Customer Satisfaction

Welcome Change

Deliver Frequently

Working Together

Motivated Team

Face to Face

Working Software

Constant Pace

Good Design

Simplicity

Self Organization

Reflect and Adjust

Agile Values

- **Individuals and interactions**

  Along with the software tools and processes, agile suggests that the people working in the processes are equally if not more important. A project can be the most successful if the people working on it are the best suited for it.

- **A working software**

  Working software is more important. It is of utmost importance according to the agile manifesto to provide the customers with working software than to have piles and piles of documentation.

Contd..

- **Customer collaboration**

  Previously there would only be a contract between the customer and the developer. So after the project would be completed it would be handed over to the customer. Many times a situation would arise that the product asked for and the product delivered was not the same. Thus agile insists on continuous delivery so that the developer and the customer are on the same page and can react to changing market conditions.

- **Responding to change**

  According to the Agile Manifesto, there might be many phases in software development when changes can be done to the product. These changes should be encouraged irrespective of the phase the project is in. This helps in achieving better goals and revised results.